UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTE À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
H. JOËL NANGA NDJANA

SYSTÈME AUXILIAIRE DE COMPENSATION DE CREUX DE TENSION

MARS 2005

RESUMÉ

Les creux de tension représentent le problème de qualité d'alimentation électrique causant les plus grandes pertes de production en industrie. La désensibilisation des équipements par la modification des topologies de convertisseurs de puissance représente une des solutions au problème.

Un système auxiliaire de compensation de creux de tension qui s'adapte aux installations existantes avec redresseur de tête en pont triphasé à diodes est développé. Ce système auxiliaire permet de désensibiliser les équipements électriques face aux creux de tension équilibrés et déséquilibrés. Il permet également d'améliorer le taux de distorsion global du courant dans la ligne par l'injection du troisième harmonique.Une version modifiée d'un algorithme basé sur la méthode Adaline permet une détection et une estimation rapide des creux de tension. Une loi de commande basée sur la compensation de la valeur moyenne de tension permet de réguler la tension du lien cc en présence de creux de tension équilibrés et déséquilibrés. Les performances de l'algorithme de détection ainsi que de la loi de commande ont été validées en simulation à l'aide du logiciel PSIM.

Le système auxiliaire de compensation a été réalisé pratiquement et des résultats expérimentaux démontrent son efficacité et sa robustesse en présence de creux de tension équilibrés et déséquilibrés.

REMERCIEMENTS

Je tiens à exprimer ma profonde reconnaissance et mes sincères remerciements :

- À Messieurs Pierre Sicard et Éloi Ngandui qui sont respectivement mon directeur et mon co-directeur de mémoire pour leur disponibilité à répondre avec exactitude aux questions soulevées par mon sujet de recherche.
- À Monsieur Sylvain Lahaie, Chercheur au Laboratoire des Technologies de l'Énergie d'Hydro-Québec pour son aide précieuse, ses encouragements et sa disponibilité lors de l'implantation. De ses connaissances et de sa longue expérience en milieu industriel dont j'ai bénéficié.
- À Hydro-Québec par le biais du Laboratoire des Technologies de l'Énergie (LTE/IREQ) qui m'a supporté financièrement lors de mes études de deuxième cycle et a assumé la totalité des coûts relatifs au matériel.
- À l'Université du Québec à Trois-Rivières par le biais de sa fondation, de laquelle, j'ai reçu deux bourses d'études supérieures.
- À Monsieur Ahmed Chériti pour le temps qu'il m'a accordé; en particulier pour les discussions qui m'ont permis de régler certaines difficultés d'ordre pratique.
- À mes parents, qui ont consenti beaucoup de sacrifices afin que je puisse faire des études en génie. À leur soutien financier, moral et spirituel tout au long de mes études.
- Enfin, à Dieu pour la bonne santé qu'il m'accordé au nom de Jésus-Christ.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTES DES SYMBOLES ET ABBRÉVIATIONS

a : Nombre complexe qui, appliqué à un vecteur, le fait tourner de $2\pi/3$.

A_{dm} : Coefficients de Fourier des composantes harmoniques de la tension redressée.

A_1 : Amplitude du fondamental d'un signal.

b(t) : Bruit aléatoire.

B : Bande passante.

B_{dm} : Coefficients de Fourier des composantes harmoniques de la tension redressée.

e_a, e_b, e_c : Source de tension triphasée.

e_d : Tension de sortie d'un redresseur triphasé à diodes.

EVV : Entraînement à vitesse variable.

f_c : Fréquence de coupure théorique.

$f_{c\,réelle}$: Fréquence de coupure réelle.

H_n : Taux de distorsion harmonique de rang n ou taux individuel d'harmonique.

I : Matrice identité.

I_A : Courant redressée sur le terminal positif d'un redresseur triphasé à diodes.

I_B : Courant redressée sur le terminal négatif d'un redresseur triphasé à diodes.

I_m : Amplitude du courant redressé.

I_o : Composante continue du courant redressé.

I_{3A} : Composante du 3e harmonique du courant du terminal positif d'un redresseur triphasé à diodes.

I_{3B} : Composante du 3e harmonique du courant du terminal positif d'un redresseur triphasé à diodes.

I_{h3} : Courant du troisième harmonique.

R_d : Résistance de la charge.

S_x : Fonction de commutation de la phase x ($x \in \{a,b \text{ et } c\}$).

T : (Opérateur) Transposition.

THD : Taux global de distorsion harmonique.

Thy$n$: Impulsion d'amorçage du thyristor n d'un pont triphasé ($n \in \{1,2,..6\}$).

U_{do} : Valeur moyenne de la tension d'un redresseur triphasé à diodes.

$U_{do\alpha}$: Valeur moyenne de la tension d'un redresseur triphasé à thyristors.

U_{d1o} : Valeur moyenne de la tension d'un redresseur triphasé à thyristors avec diode de roue libre.

U_{doA}, U_{doB}, U_{doC} : Contribution de la phase A, B et C à la valeur moyenne de la tension d'un redresseur triphasé à diodes.

$U_{do\alpha A}$, $U_{do\alpha B}$, $U_{do\alpha C}$: Contribution de la phase A, B et C à la valeur moyenne de la tension d'un redresseur triphasé à thyristors.

V_A : Tension redressée sur le terminal positif d'un redresseur triphasé.

V_B : Tension redressée sur le terminal négatif d'un redresseur triphasé.

V_a, V_b, V_c : Tensions d'une source triphasée.

V_d : Composante directe de la tension.

V_H : Largeur de l'hystérésis.

V_i : Composante inverse de la tension.

V_o : Composante homopolaire de la tension.

V_{LL} : Amplitude de la tension ligne-ligne.

V_{LN} : Amplitude de la tension ligne-neutre.

V_m : Amplitude de la tension ligne-neutre.

K : Matrice des gains d'adaptation des coefficients de Fourier.

K_{T1} : Rapport de transformation du transformateur T_1.

L_d : Inductance de lissage.

n : (Indice) Rang harmonique.

P : Inverse de la matrice d'autocorrelation du vecteur X.

PIB : Point inférieur de basculement.

PSB : Point supérieur de basculement.

W : Matrice des poids.

X : Vecteur des coefficients de la série de Fourier d'un signal.

X_n, Y_n : Coefficients de la série de Fourier d'un signal.

X_1, Y_1 : Coefficients de Fourier du fondamental.

ΔU_{do} : Diminution de la tension moyenne du lien cc.

α : Angle d'amorçage d'un thyristor.

λ : Facteur d'oubli.

θ_n : Instant de commutation n.

ϕ_1 : Phase du fondamental d'un signal.

τ_v : Taux de déséquilibre de tension.

μ : Angle d'empiètement.

ω, ω_o : Pulsation de la tension.

CHAPITRE 1 : INTRODUCTION

1.1 Introduction

La qualité de l'alimentation électrique est un sujet qui a connu un regain d'intérêt ces deux dernières décennies. Ce problème est caractérisé par les variations que peuvent subir la tension, le courant ou la fréquence. Ce regain d'intérêt peut s'expliquer pour certaines raisons telles que :

- Les équipements électriques modernes intègrent de plus en plus des circuits de commande basés sur les microprocesseurs; ceux-ci sont plus sensibles aux variations de l'alimentation électrique.
- L'introduction massive des semi-conducteurs de puissance dans les procédés industriels qui a contribué à augmenter leur efficacité. Par contre, cette utilisation de semi-conducteurs a contribué à augmenter le niveau de pollution harmonique. Selon EPRI (Electric Power Research Institute), aux États-Unis en 1985, 20% des charges électriques étaient électroniques. Ce nombre a connu une augmentation d'approximativement 50 à 60% en 2000 [1].
- L'impact économique des problèmes reliés à cette problématique est important [2].

Par exemple, des incidents tels que des défauts (court-circuits) dans les installations ou un brusque démarrage d'une machine tournante à forte puissance peuvent causer une chute soudaine de la tension. On nommera ce type d'incident : *creux de tension*. Les creux de tension constituent le problème de la qualité de l'alimentation électrique le plus important non seulement par son occurrence mais par son impact économique. Selon [3], 68% des perturbations enregistrées étaient

des creux de tension et étaient la principale cause des pertes de production. Ces perturbations nuisent au bon fonctionnement des équipements stratégiques des entreprises industrielles tels que les automates programmables (PLC) qui contrôlent des procédés ou encore les entraînements à vitesse variable.

Dans les sections qui suivent nous définirons quelques-uns des problèmes reliés à la qualité de l'alimentation électrique. Nous parlerons des effets des creux de tension sur les entraînements à vitesse variable ainsi que de quelques solutions existantes.

1.2 Quelques problèmes reliés à la qualité de l'alimentation électrique

Sauf exception, les tensions d'un réseau électrique constituent un système alternatif triphasé, dont la fréquence de base est de 50 Hz (en Europe et en Afrique) ou de 60 Hz (en Amérique du Nord)[1]. Les paramètres caractéristiques d'un tel système sont les suivants :

- la fréquence,
- l'amplitude des trois tensions,
- la forme d'onde qui doit être la plus proche possible d'une sinusoïde,
- la symétrie du système triphasé, caractérisée par l'égalité des modules des trois tensions et de leur déphasage relatif.

Les problèmes reliés à la qualité de l'alimentation électrique se manifestent sous différentes formes affectant un ou plusieurs des quatre paramètres précédemment définis. On peut donc citer :

[1] Sans perte de généralité, une fréquence de base de 60Hz est posée dans l'ensemble de ce travail

Les fluctuations de la fréquence à 60 Hz : elles sont rares et ne sont observées que lors de circonstances exceptionnelles.

Les variations de l'amplitude : il ne s'agit pas des variations lentes de tension, mais de variations rapides de tension ou de creux de tension. Les creux de tension peuvent être de forme régulière ou non.

La déformation de la forme d'onde de la tension ou du courant : cette onde n'est alors plus sinusoïdale, et peut être considérée comme représentable par une onde fondamentale à 60 Hz, associée soit à des harmoniques de fréquence multiple entier de 60 Hz, soit même parfois à des ondes de fréquence quelconque.

La dissymétrie du système triphasé, que l'on appelle déséquilibre.

On peut, en plus, mentionner un type particulier de perturbations difficile à classer puisqu'il concerne tout à la fois l'amplitude et la forme d'onde : ce sont les variations transitoires d'amplitudes (*transients* en anglais) dont la durée est inférieure à 10 ms.

1.2.1 Harmoniques

Un des problèmes prédominant dans le domaine de la qualité de l'onde est la pollution harmonique [4]. La prolifération des équipements électriques utilisant des convertisseurs statiques a entraîné ces dernières années une augmentation sensible du niveau de pollution harmonique des réseaux électriques. Ces équipements électriques sont considérés comme des charges non linéaires émettant des courants harmoniques dont les fréquences sont des multiples entiers de la fréquence fondamentale, ou parfois à des fréquences quelconques. Le passage de ces courants harmoniques dans les impédances du réseau électrique peut entraîner des tensions harmoniques aux points de raccordement (PCC) et alors polluer les consommateurs alimentés par le même réseau électrique. De nombreux effets des

harmoniques sur les installations et les équipements électriques peuvent être cités. Les effets les plus importants sont les échauffements (conducteurs, transformateurs et machines électriques), les défauts de fonctionnement de certains équipements électriques et le risque d'excitation de résonance. Différentes grandeurs sont définies pour chiffrer les perturbations harmoniques. Parmi les plus utilisées on peut citer : le *taux global de distorsion harmonique* (THD : *Total Harmonic Distorsion*) qui mesure l'importance des harmoniques par rapport au fondamental et *le taux de distorsion harmonique de rang n (ou taux individuel d'harmonique)* qui mesure l'importance de chacun des harmoniques par rapport au fondamental. Nous le notons H_n. Ils s'expriment respectivement comme suit :

$$THD = \frac{\sqrt{\sum_{n=2}^{\infty} I_n^2}}{I_1} \tag{1.1}$$

$$H_n = \frac{I_n}{I_1} \tag{1.2}$$

n est le rang harmonique,

I_1 est l'amplitude du fondamental,

I_n est l'amplitude du n^{ieme} harmonique.

1.2.2 Déséquilibres

Dans un système triphasé, on parle de déséquilibre lorsque l'égalité des modules des trois tensions ou de leur déphasage relatif n'est plus vérifiée. Le déséquilibre en tension est caractérisé par le taux de déséquilibre de tension τ_v donné par le rapport des amplitudes des tensions inverse et directe :

$$\tau_v = \frac{V_i}{V_d} \tag{1.3}$$

$$\begin{bmatrix} \overline{V_o} \\ \overline{V_d} \\ \overline{V_i} \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix}\begin{bmatrix} \overline{V_a} \\ \overline{V_b} \\ \overline{V_c} \end{bmatrix} \tag{1.4}$$

où $a = -\frac{1}{2} + \frac{1}{2} j\sqrt{3}$

V_o : composante homopolaire de la tension

V_d : composante directe de la tension

V_i : composante inverse de la tension

Le déséquilibre de tension peut survenir pour plusieurs raisons; par exemple, la transposition non complétée des lignes de transmission, des charges déséquilibrés, la connexion ouverte d'un transformateur delta, la prolifération des charges non linéaires et monophasées. Le déséquilibre de tension peut causer des effets indésirables dans le réseau, particulièrement lorsque des charges sensibles y sont connectées. Par ailleurs le déséquilibre peut contribuer à empirer la qualité de l'alimentation électrique lorsque des charges non linéaires sont connectées au réseau. Par exemple, les convertisseurs statiques alimentés par une alimentation déséquilibrée produisent des harmoniques basses fréquences non caractéristiques [6]. Outre les effets classiques des harmoniques, ces fréquences non caractéristiques peuvent conduire, dans certains cas, au blocage de la commande [7]. Les machines ca sujets à un déséquilibre peuvent générer une grande composante inverse du courant due à la faible composante inverse de l'impédance, causant ainsi des pertes dans la machine et la réduction du couple [5].

1.2.3 Creux de tension

Dans la littérature il existe plusieurs définitions d'un creux de tension. Citons selon nous quelques unes des plus intéressantes :

1. Un creux de tension est une baisse momentanée de l'amplitude de la tension pendant quelques cycles [8].
2. Un creux de tension est une diminution momentanée de la valeur efficace de la tension ayant une durée allant d'un demi cycle à 1 minute [9].
3. Un creux de tension est une baisse de la tension avec une durée variant entre un cycle et quelques secondes [10].
4. Selon le standard IEEE 1159-1995, un creux de tension est une baisse de 10% à 90% de la valeur efficace de la tension variant d'un demi-cycle à une minute.

On caractérise les creux de tension par leur amplitude et leur durée. Il est important de souligner la nuance qui existe lorsqu'on parle de l'amplitude d'un creux. Cette caractéristique peut avoir deux définitions. Selon certains auteurs, l'amplitude d'un creux désigne la tension résiduelle et selon d'autres elle désigne la diminution de la tension elle-même. Dans la suite de ce travail, nous adopterons la deuxième définition. Par exemple, lorsque nous parlerons d'un creux de tension de 30%, cela signifiera que la tension à subit une baisse de 30% de la tension nominale. La figure 1.1 présente l'exemple d'un creux de tension dû à un court-circuit.

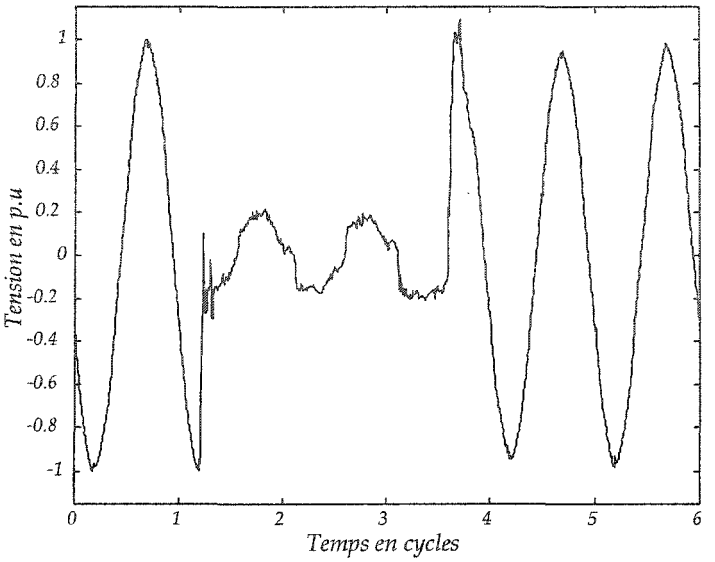


Figure 1.1 : Exemple d'un creux dû à un court-circuit [3].

Les creux de tension sont susceptibles de perturber le fonctionnement de certains équipements électriques. Parmi ces équipements, on distingue les équipements de traitement de données (ordinateurs), les entraînements à vitesse variable (EVV), les automates programmables, les machines à courant alternatif et à courant continu [8]. Cependant des études démontrent que ces différents types d'équipements présentent des sensibilités (tolérances) très différentes face aux creux de tension, d'où la difficulté de développer un standard qui couvre tous les équipements industriels. La courbe CBEMA développée par la *Computer Businness Equipment Manufacturers Association* (web : www.itic.org) a été largement utilisée comme référence pour déterminer la sensibilité des équipements industriels face aux creux de tension. Cette courbe fût initialement bâtie pour les équipements de traitement de données. La courbe CBEMA a été révisée par la *Information Technology Council (ITIC)* donnant naissance à la courbe ITI présentée à la figure 1.2.

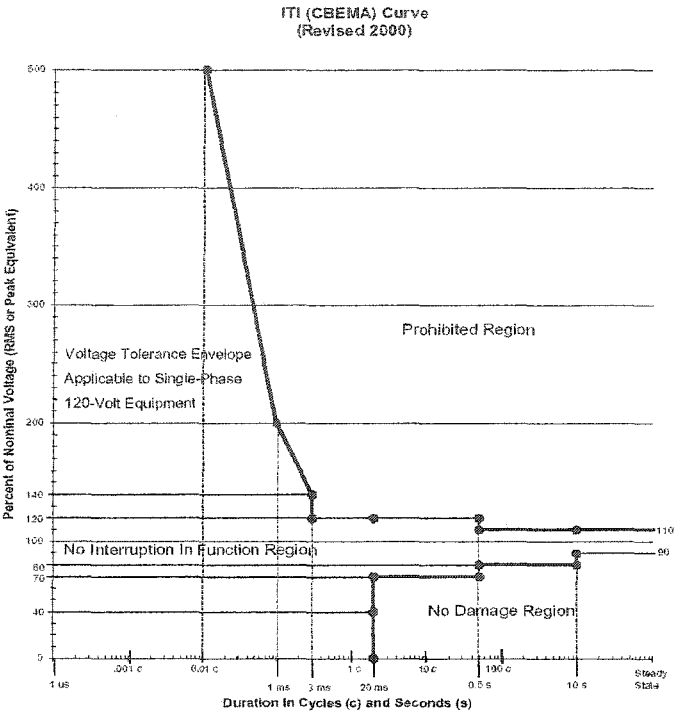


Figure 1.2 : Courbe ITI[2].

Il existe également un autre standard beaucoup plus récent développé par EPRI PEAC Corporation connu sous le nom de SEMI F47 pour « *Specification for Semiconductor Processing Equipment Voltage Sag Immunity* ». EPRI PEAC Corporation est une société d'ingénierie qui aide ses clients à comprendre, diagnostiquer, résoudre et à prévenir des problèmes liés à la qualité de l'alimentation électrique et l'efficacité énergétique. Elle travaille également dans la production distribuée.

SEMI F47 est un standard qui définit un seuil au delà duquel un dispositif à semiconducteurs peut fonctionner sans interruption. Il définit ainsi certaines spécifications électriques à respecter par les équipements électriques. L'industrie des semiconducteurs est reconnue comme étant très exigeante en ce qui concerne la qualité de son alimentation électrique à cause de la sensibilité de ses

[2] Source : www.itic.org.

équipements et ses procédés de contrôle. Cette industrie est particulièrement vulnérable aux creux de tension. SEMI F47 permet de définir la capacité de ses équipements électriques à passer à travers un creux de tension. Les creux de tension et leurs durées tel que définis par SEMI F47 sont présentés au tableau 1-1.

Tableau 1-1 : Standard SEMI F47

Type de creux de tension	Amplitude résiduelle (%)	Durée	
		(seconde)	(cycles)
Creux de tension monophasé et biphasé	50	0.05	3
	50	0.2	12
	70	0.2	12
	70	0.5	30
	80	0.5	30
	80	1	60

La courbe SEMI F47 est présentée à la figure 1.3.

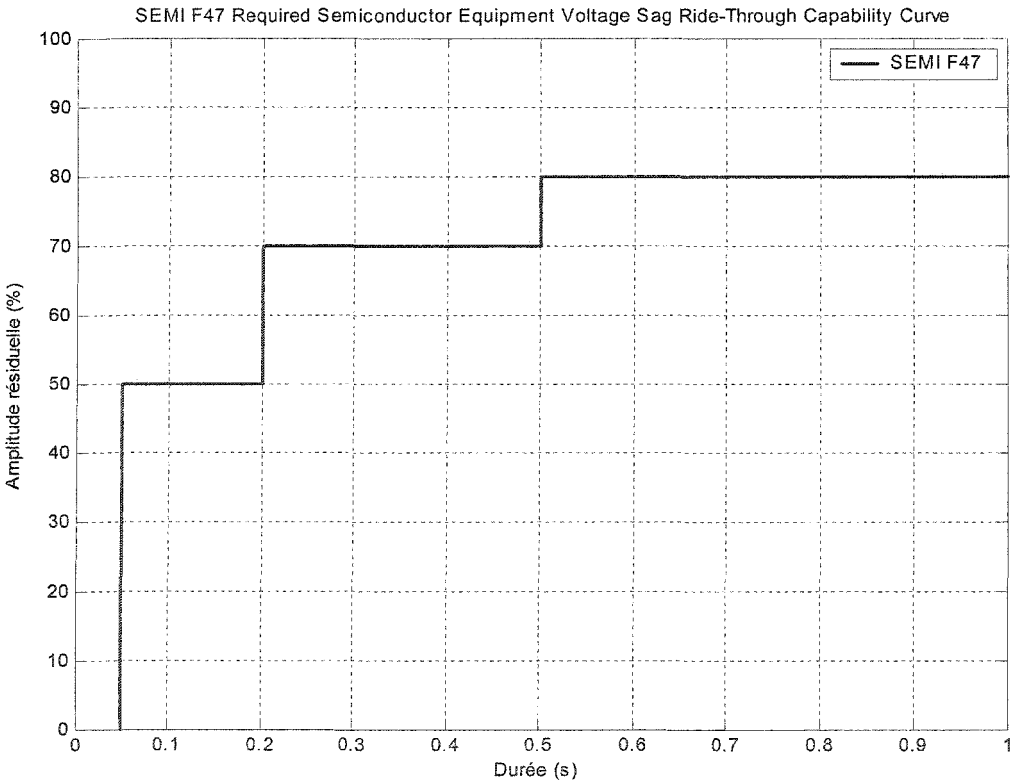


Figure 1.3 : Courbe SEMI F47.

Selon le standard SEMI F47, lors d'un creux de tension, l'équipement électrique doit continuer fonctionner sans interruption si on est dans les conditions de la zone située au dessus de la courbe (voir figure 1.3). Dans le contexte du standard SEMI F47, interruption veut dire sans assistance extérieure, par exemple d'un opérateur ou encore un fonctionnement hors des limites normales d'opération.

Ce standard international a premièrement été développé pour satisfaire les besoins de l'industrie des semiconducteurs. Allant dans le même sens que les standards (courbes) développés par d'autres organisations, SEMI F47 s'impose comme un standard générique qu'on peut appliquer pour d'autres types d'équipements, autres que ceux qu'on retrouve dans l'industrie des semiconducteurs. Il se présente également comme un outil intéressant lors de la conception pour définir les spécifications ou encore lors de l'amélioration d'un équipement ou d'un procédé. Nous l'utiliserons dans la suite de ce travail pour valider les performances du système auxiliaire de compensation que nous présentons.

1.3 Effets des creux de tension sur les entraînements à vitesse variable

Les EVV sont connus pour être très sensibles aux creux de tension [11]. La figure 1.4 montre le schéma typique d'un entraînement à vitesse variable qui comprend un redresseur triphasé à diodes, un lien continu appelé aussi lien cc et un onduleur.

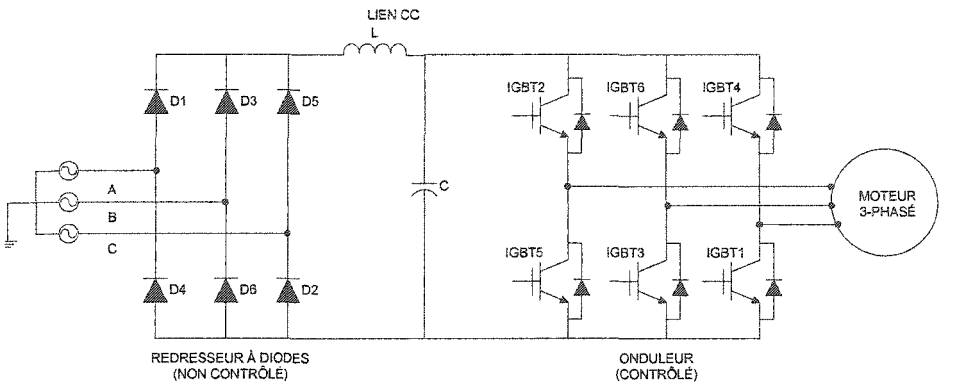


Figure 1.4 : Topologie typique d'un entraînement à vitesse variable

Plusieurs méthodes permettent de classifier les creux de tension. Une des plus complètes a été présentée par l'auteur Math Bollen en [3]. Il identifie sept types de creux. Cette classification sera présentée dans le chapitre suivant. Cependant en [11], il est démontré que cette classification n'est pas nécessaire pour déterminer le comportement des EVV en présence de creux de tension. On peut tout simplement parler de creux monophasé (sur une seule phase), biphasé (sur deux phases) et triphasé (sur les trois phases).

Les creux de tension entraînent une diminution du lien cc qui peut entraîner des fluctuations du couple ou de la vitesse ou encore un arrêt total de la machine et voire de l'unité de production [11]. Dans l'industrie textile et papetière, le problème est criant. Un creux de tension de brève durée peut causer une fluctuation de la vitesse qui peut endommager le produit fini.

Des études réalisées pendant une période de 17 mois sur deux sites industriels avec des EVV ont mené les chercheurs à la conclusion que des creux de tension ayant une durée de 12 cycles (0.2 seconde) et plus et une amplitude de plus de 20% conduiront au délestage de l'EVV [9], [10].

1.4 Quelques solutions existantes pour la compensation des creux de tension

Plusieurs méthodes sont proposées dans la littérature pour préserver les équipements des creux de tension. Elles peuvent être classifiées en trois catégories : *l'installation des équipements de compensation, l'utilisation d'une alimentation de soutien* et *la modification des topologies des convertisseurs*. La liste de solutions que nous présentons n'est pas exhaustive, mais regroupe les différentes solutions les plus couramment rencontrées.

1.4.1 Utilisation des équipements de compensation

Ces équipements sont généralement installés au point de couplage commun (PCC) ou avant le PCC afin de compenser des creux de tension.

1.4.1.1 Compensateur série de tension

Il s'agit d'une source contrôlée de tension placée en série entre le réseau et le PCC (voir figure 1.5). Ce type d'équipements permet de désensibiliser les charges sensibles aux creux de tension. À l'apparition d'un creux de tension, il injecte sur la ligne, une tension alternative d'amplitude, de phase et de fréquence contrôlées. Il peut être aussi contrôlé de façon à empêcher la circulation des harmoniques des charges non-linéaires vers le réseau et de compenser les déséquilibres qui peuvent survenir dans le réseau.

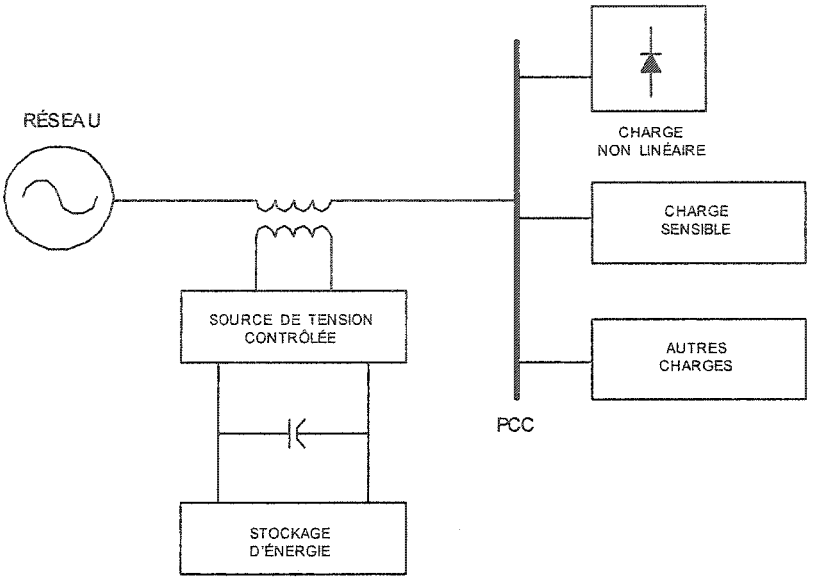


Figure 1.5 : Compensateur série de tension

Le terme DVR (*Dynamic Voltage Restorer*) est plus souvent utilisé à la place de compensateur série [3]. La puissance des DVR disponibles sur le marché est de quelques MVA et permettent de compenser des creux de tension jusqu'à 50% d'amplitude et pour une durée de quelques centaines de millisecondes [2]. Le principal inconvénient de ce type d'équipement est qu'il ne protège pas les charges sensibles contre les creux de tension causés par un défaut (par exemple un court-circuit) à l'intérieur de l'installation.

1.4.1.2 Compensateur combiné parallèle et série (UPQC)

Le compensateur combiné parallèle et série est aussi appelé *Unified Power Quality Conditioner* (UPQC). C'est une configuration hybride qui résulte de l'association des deux filtres actifs parallèle et série, comme le montre la figure 1.6. Profitant des avantages des deux filtres actifs, l'UPQC assure un courant et une tension sinusoïdaux à la charge et la protégeant aussi d'éventuels creux de tension. Ce dispositif permet aussi de compenser la puissance réactive.

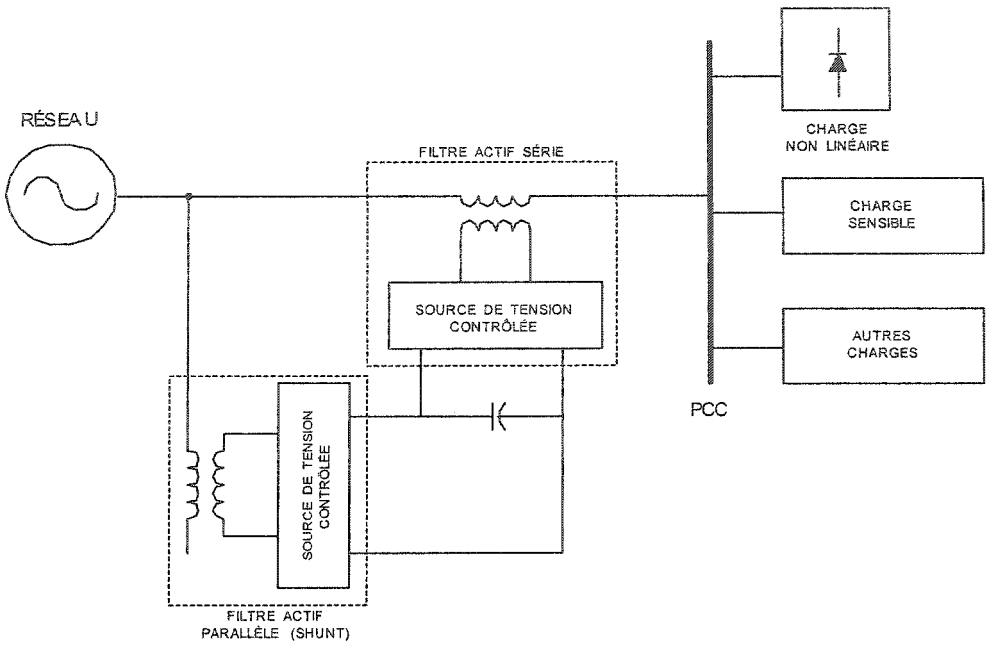


Figure 1.6 : Structure générale d'un UPQC

1.4.2 Utilisation des alimentations de soutien

Cette solution préconise l'utilisation des alimentations de type UPS (*Uninterruptible Power Supply*). Sa configuration de base est présentée à la figure 1.7.

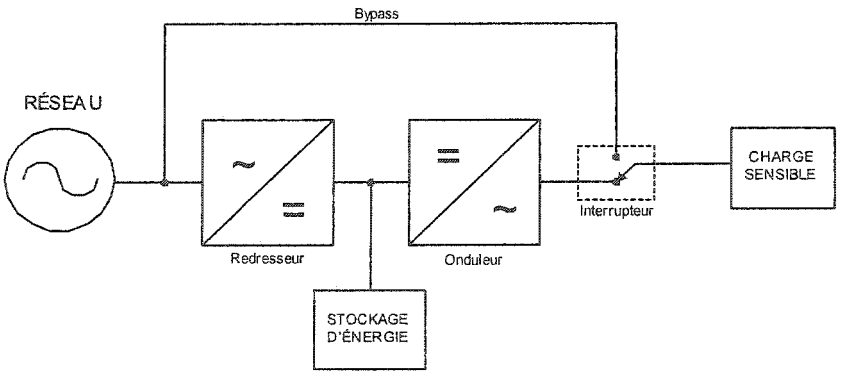


Figure 1.7 : Configuration typique d'un UPS

Dans les UPS qu'on retrouve sur le marché, le stockage d'énergie se fait à l'aide de batteries [3].

En l'absence de creux de tension, la charge est alimentée via l'UPS qui tire son énergie du réseau. La conception de l'UPS est faite de sorte que la tension de son lien cc (entre le redresseur et l'onduleur) soit légèrement supérieure à celle des batteries; ainsi elles demeurent en attente. En cas de dysfonctionnement de l'onduleur, à l'aide de l'interrupteur la charge sera connectée directement au réseau en court-circuitant l'UPS.

En présence de creux de tension, l'énergie stockée dans les batteries servira à maintenir le niveau de tension du lien cc constant.

L'UPS est une solution attrayante de par la simplicité de son fonctionnement et de son contrôle. En revanche, il présente deux principaux inconvénients : l'utilisation de deux convertisseurs supplémentaires et l'utilisation des batteries. L'utilisation des convertisseurs entraîne des pertes d'énergie supplémentaires dues à la double conversion et contribue à augmenter le taux global de distorsion du courant dans la ligne car ces convertisseurs constituent des charges non-linéaires. Les batteries elles, ont souvent besoin de vérification pour s'assurer qu'elles fonctionneront bien en présence de creux de tension.

1.4.3 Modification des topologies des convertisseurs

Cette solution est très utilisée pour les entraînements à vitesse variable. Dans la littérature, on rencontre plusieurs types de modifications dont le but principal est de réguler la tension du lien cc à l'intérieur des limites raisonnables pendant un creux de tension.

1.4.3.1 Utilisation d'élévateur de tension

Un élévateur de tension est ajouté entre le redresseur à diodes et le condensateur du lien cc. Le schéma du montage est présenté à la figure 1.8. Son rôle est maintenir la tension du lien cc dans les limites acceptables pour un fonctionnement normal durant un creux de tension. Cette régulation est possible grâce à la variation du rapport cyclique de l'interrupteur Q.

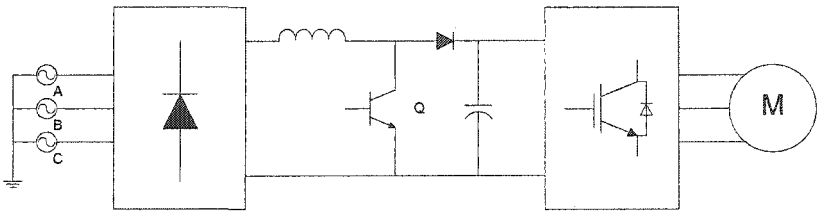


Figure 1.8 : EVV avec un élévateur de tension.

1.4.3.2 Utilisation d'un redresseur parallèle

Dans ce cas un redresseur triphasé à diodes suivi d'un élévateur de tension est raccordé en parallèle avec un EVV standard. Le schéma du montage est présenté à la figure 1.9. Le convertisseur raccordé en parallèle permet de réguler la tension du lien cc à l'apparition d'un creux de tension.

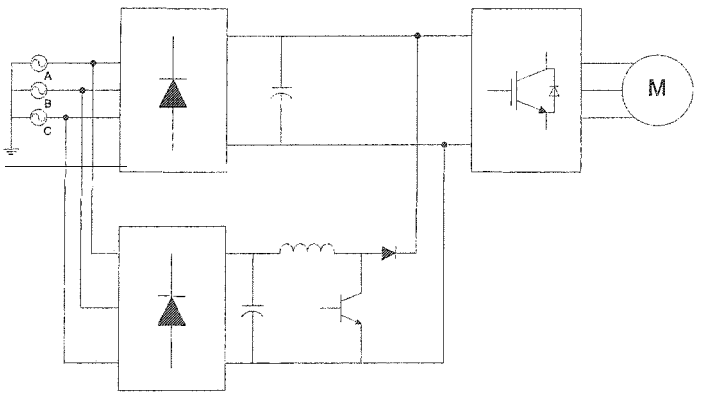


Figure 1.9 : EVV avec un redresseur parallèle.

Duran-Gomez et *al.* présentent en [10] une autre variante de cette topologie où le redresseur auxiliaire à diodes double alternance est remplacé par un redresseur simple alternance.

1.4.3.3 Utilisation d'un redresseur actif

La topologie est semblable à celle d'un EVV standard à seule différence que le redresseur à diodes est substitué par un redresseur actif à IGBT. La figure 1.10 présente le schéma du montage. Une commande appropriée du pont (IGBT) permet de réguler la tension du lien cc en cas de perturbations à la source.

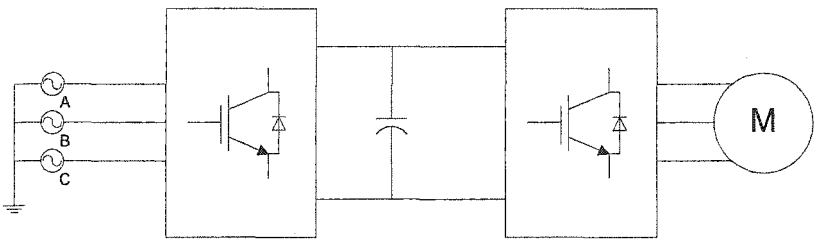


Figure 1.10 : EVV avec un redresseur actif

1.4.4 Conclusion

Les procédés de fabrication, avec leur grand nombre d'entraînements à vitesse variable, sont sensibles aux défauts électriques de même qu'aux creux de tension. Les besoins importants en énergie électrique imposent une utilisation optimale des ressources disponibles et des solutions spéciales pour améliorer la qualité de l'alimentation électrique. Nous avons présenté quelques solutions existantes et qui protègent les équipements électriques, principalement les EVV des creux de tension. À cause de l'impact économique des creux de tension dans certaines industries (textiles, papetières, etc.), il est de plus en plus important d'apporter des solutions viables et surtout moins dispendieuses.

1.5 Objectifs et organisation du mémoire

Il est proposé de développer un système auxiliaire de compensation de creux de tension qui s'adapte aux installations existantes avec redresseur de tête en pont triphasé à diodes et permettant de désensibiliser les équipements électriques face aux creux de tension équilibrés et déséquilibrés. Ce système permettra entre autre d'améliorer le taux de distorsion global du courant dans la ligne grâce à la méthode d'injection du troisième harmonique.
Pour atteindre cet objectif, plusieurs problématiques seront abordées et traitées telles que : (a) modification et modélisation de la topologie des convertisseurs; (b) détection rapide et estimation de creux de tension; (c) conception d'une loi de compensation de creux de tension rapide et robuste; (d) réalisation d'un prototype.

Ce mémoire est organisé de la façon suivante :

Au chapitre 2, une présentation des différents types de creux de tension selon l'auteur Math Bollen est faite. Les effets des transformateurs sur le type de creux de tension sont étudiés. Le concept des fonctions de commutation est utilisé pour analyser et déterminer des équations pour l'estimation de la valeur moyenne de la tension redressée, en présence de creux de tension équilibrés et déséquilibrés pour deux convertisseurs : un redresseur triphasé à diodes et redresseur triphasé à thyristors. Des résultats d'une application numérique sont présentés afin de valider la méthode d'analyse choisie.

Au chapitre 3, la détection et l'estimation des creux de tension basées sur la méthode Adaline sont présentées. L'algorithme de cette méthode est présenté ainsi que ses performances pour la détection et l'estimation des creux de tension.

Au chapitre 4, une description détaillée des deux systèmes auxiliaires et leur principe de fonctionnement sont exposés : le système de compensation des creux de tension et le système de compensation des harmoniques. D'après l'analyse faite au chapitre 2, une loi de compensation de creux de tension équilibrés et déséquilibrés est élaborée. Des résultats de simulation obtenus avec le logiciel PSIM sont également présentés.

Au chapitre 5, une présentation sommaire de l'environnement expérimental est faite. L'implantation des algorithmes de détection de creux de tension et de la loi de compensation et la conception des divers circuits électroniques de commande sont exposées. Les résultats expérimentaux sont aussi présentés.

Des conclusions sont tirées et quelques recommandations sont proposées au chapitre 6.

CHAPITRE 2 : EFFETS DES CREUX DE TENSION SUR LES CONVERTISSEURS CA-CC

2.1 Introduction

Le nombre et la puissance unitaire des redresseurs à diodes et à thyristors ne cessent de croître dans les réseaux électriques. Il est estimé que dans un avenir proche, 60% de l'énergie électrique produite sera traitée par les convertisseurs statiques [12]. Ces convertisseurs ont été très largement utilisés dans les entraînements à vitesse variable (EVV), soudage, en électrolyse, etc. Ils sont requis partout où la conversion d'énergie ca/cc est requise et sont utilisés dans une grande variété d'applications [13]. Les entraînements à vitesse variable (EVV) apparaissent d'ailleurs être plus sensibles aux creux de tension que les équipements de traitement de données (ordinateurs) [10].

Dans les EVV, les creux de tension se manifestent principalement par la diminution de la tension dans le lien cc [3]. Dans ce chapitre, nous nous intéresserons donc à la variation de la valeur moyenne de la tension redressée en fonction des conditions de l'alimentation. Le concept des fonctions de commutation sera utilisé. Ce concept a été utilisé pour calculer le contenu harmonique côtés alternatif et continu d'un redresseur à thyristors [12], [13] et la bonne précision des résultats obtenus est démontrée.

2.2 Classification des creux de tension

Un creux de tension est défini comme la réduction momentanée de la valeur efficace de la tension nominale pour une durée allant de 0.5 cycle à une minute [9].

Les creux de tension sont principalement causés par des défauts tels que les courts-circuits et le démarrage de gros moteurs à induction [3]. Les creux de tension sont la principale cause des perturbations que subissent les entraînements à vitesse variable, les ordinateurs et les contrôleurs de procédés industriels. En [14], une étude de cas réalisée sur le monitoring des perturbations reliées à la qualité de l'alimentation électrique a révélé que 68% des perturbations enregistrées sont des creux de tension et qu'ils représentent la principale cause des pertes de production. Afin d'analyser les effets des creux de tension sur les convertisseurs ca/cc, il est nécessaire de les caractériser et de les classifier.

Il existe deux principaux paramètres qui caractérisent un creux de tension [15]. Ces paramètres sont : l'amplitude du creux ou profondeur du creux qui définit la diminution de la tension efficace et la durée qui définit le temps pendant lequel se produit le creux. Ces deux paramètres sont généralement représentés par un graphique à deux dimensions. La figure 2.1 illustre l'amplitude résiduelle de la tension pendant un creux. La durée du creux est exprimée en cycle; un cycle correspond à 16.67 ms dans un réseau de 60Hz.

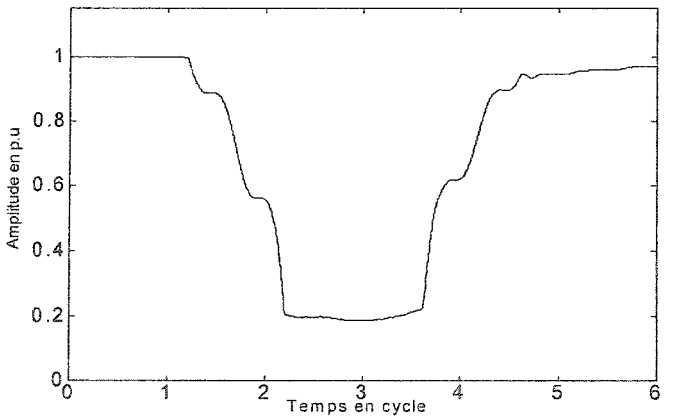


Figure 2.1 : Amplitude du fondamental en fonction du temps lors d'un creux de tension

Ce mode de représentation est simpliste car il ne tient pas compte de l'asymétrie entre les tensions des phases et des sauts de phase qui peuvent survenir tel que décrit par Bollen en [3]. Aussi, il ne tient pas compte de la nature non sinusoïdale de la tension lors d'un creux.

On peut distinguer trois grands types de creux : i) les creux monophasés qui surviennent entre une phase et la terre; ii) les creux biphasés qui surviennent entre deux phases et; iii) les creux triphasés qui surviennent sur les trois phases. Il faut signaler que les deux derniers peuvent être symétriques ou asymétriques (respectivement même amplitude ou amplitude différente sur chacune des phases affectées). Les creux monophasés sont les plus fréquents tandis que les creux biphasés et triphasés le sont moins; soit moins 20% du temps [15]. Selon [16] approximativement 66% des creux de tension sont monophasés et les types biphasés représentent 16%.

Bollen [3] présente une classification beaucoup plus complète qui regroupe quatre types de base que sont les types A, B, C et D. Pour les définitions qui suivent, nous considérons un système triphasé de tensions exprimées en p.u.

Creux de type A

Ce type est causé par un défaut phase-terre sur chacune des trois phases. Il possède une faible occurrence. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= V \\ V_b &= -\frac{1}{2}V - j\frac{\sqrt{3}}{2}V \\ V_c &= -\frac{1}{2}V + j\frac{\sqrt{3}}{2}V \end{aligned} \qquad (2.1)$$

Creux de type B

Ce type est causé par un défaut phase-terre sur une phase. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= V \\ V_b &= -\frac{1}{2} - j\frac{\sqrt{3}}{2} \\ V_c &= -\frac{1}{2} + j\frac{\sqrt{3}}{2} \end{aligned} \tag{2.2}$$

Creux de type C

Ce type est causé par un défaut phase-phase-terre. Pour ce type de creux, un saut de phase est observé. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= 1 \\ V_b &= -\frac{1}{2} - j\frac{\sqrt{3}}{2}V \\ V_c &= -\frac{1}{2} + j\frac{\sqrt{3}}{2}V \end{aligned} \tag{2.3}$$

Creux de type D

Ce type est causé par un défaut phase-phase-terre. La charge est connectée en delta et les trois phases se trouvent diminuées. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= V \\ V_b &= -\frac{1}{2}V - j\frac{\sqrt{3}}{2} \\ V_c &= -\frac{1}{2}V + j\frac{\sqrt{3}}{2} \end{aligned} \tag{2.4}$$

La figure 2.2 montre les quatre types de creux de base précédemment définis.

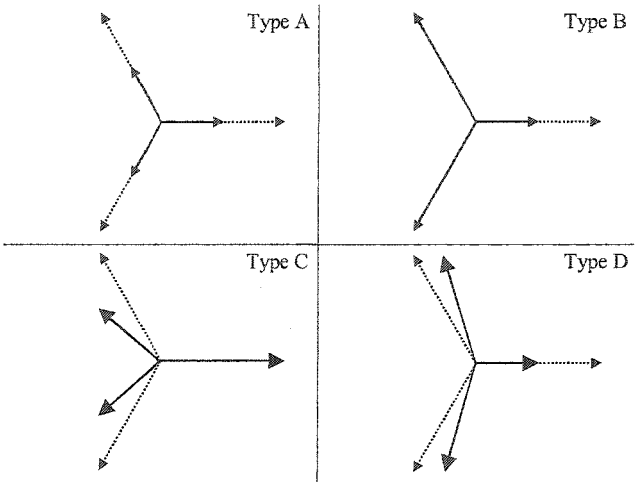


Figure 2.2 : Types de creux de tension de base

Les défauts biphasés conduisent à trois types de creux additionnels [3]. Il s'agit des creux de types E, F et G.

Creux de type E

Ce type est causé par un défaut biphasé. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= 1 \\ V_b &= -\frac{1}{2}V - j\frac{\sqrt{3}}{2}V \\ V_c &= -\frac{1}{2}V + j\frac{\sqrt{3}}{2}V \end{aligned} \qquad (2.5)$$

Creux de type F

Ce type est causé par un défaut biphasé dans une configuration où la charge est connectée en delta. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= V \\ V_b &= -\frac{1}{2}V - j\sqrt{3}\left(\frac{1}{3}+\frac{1}{6}V\right) \\ V_c &= -\frac{1}{2}V + j\sqrt{3}\left(\frac{1}{3}+\frac{1}{6}V\right) \end{aligned} \tag{2.6}$$

Creux de type G

Ce type est causé par un défaut biphasé. Il peut être défini par l'ensemble d'équations suivant :

$$\begin{aligned} V_a &= \frac{2}{3}+\frac{1}{3}V \\ V_b &= -\frac{1}{3}-\frac{1}{6}V - j\frac{\sqrt{3}}{2}V \\ V_c &= -\frac{1}{3}-\frac{1}{6}V + j\frac{\sqrt{3}}{2}V \end{aligned} \tag{2.7}$$

La figure 2.3 montre les trois types de creux précédemment définis.

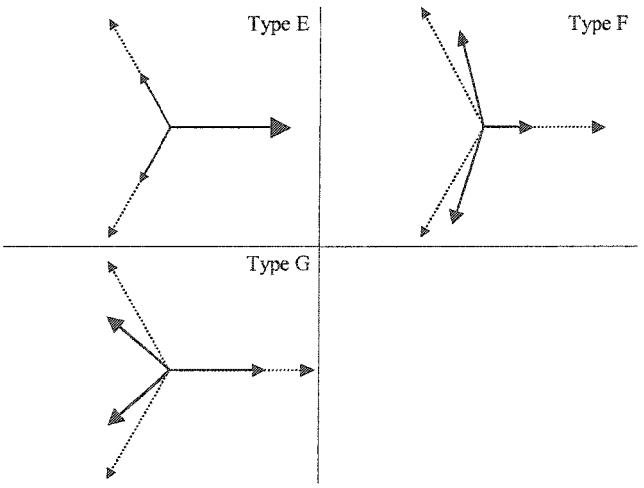


Figure 2.3 : Trois types de creux de tension dus aux défauts biphasés

2.3 Effets des transformateurs sur les creux de tension

Lors d'un creux de tension, la tension présente à un bus dépend de la distance entre le défaut et le bus et du type de connexion du transformateur [15]. Dans cette section nous présentons l'effet des transformateurs sur les creux de tension tel que perçus par la charge. Plusieurs creux de tension sont causés par des défauts monophasés (type B). Ils sont les plus fréquents tandis que les défauts biphasés (type C) et triphasés (type A) le sont moins; soit moins de 20% du temps [15].

Nous considérons dans cette étude, deux types de creux de tension : les creux monophasés et biphasés. Trois types de transformateurs sont couramment utilisés; il s'agit d'un transformateur étoile-étoile (Y-Y), d'un transformateur triangle-triangle (Δ-Δ) et d'un transformateur étoile-triangle (Y-Δ). Cette étude consiste à mesurer les tensions de lignes v_{ab}, v_{bc} et v_{ca} au secondaire des différents transformateurs en présence de creux de tension de type B et C. Les différentes charges sont équilibrées.

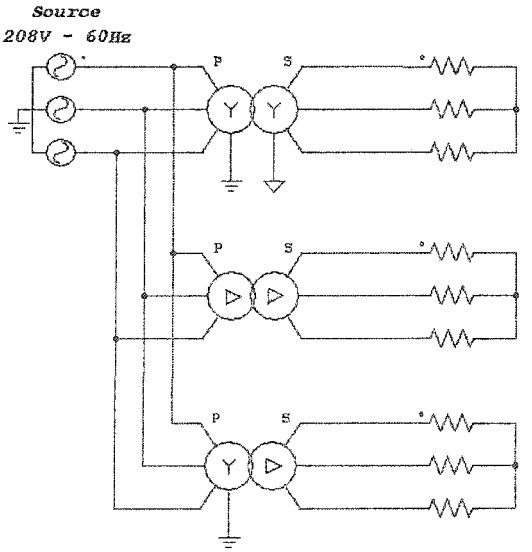


Figure 2.4 : Schéma d'étude de l'effet des transformateurs sur les types de creux de tension

Les simulations faites à l'aide du logiciel PSIM nous donnent les résultats présentés aux tableaux 2-1 et 2-2. Seul l'effet sur les amplitudes des tensions de ligne est présenté. L'effet des transformateurs sur les phases (saut de phase) n'est pas étudié car dans la suite de ce travail le saut de phase est négligé.

Il est possible de façon analytique de vérifier l'effet des transformateurs sur les creux de tension. Nous avons pu verifier la concordance des résultats des tableaux 2-1 et 2-2 et ceux de l'approche analytique présentée au chapitre 4 de la référence [35].

Tableau 2-1 : Tensions de ligne au secondaire pour les creux de tension monophasés

	Amplitude	*Transformateur*	*Tensions de ligne au secondaire du transformateur (p.u)*		
			Vab	*Vbc*	*Vca*
Type B	10%	Y-Y	0,95	1,00	0,95
		Δ-Δ	0,95	1,00	0,95
		Y-Δ	0,93	0,98	0,98
	30%	Y-Y	0,86	1,00	0,86
		Δ-Δ	0,86	1,00	0,86
		Y-Δ	0,80	0,95	0,95
	60%	Y-Y	0,72	1,00	0,72
		Δ-Δ	0,72	1,00	0,72
		Y-Δ	0,60	0,91	0,91
	90%	Y-Y	0,61	1,00	0,61
		Δ-Δ	0,61	1,00	0,61
		Y-Δ	0,40	0,88	0,88
	100%	Y-Y	0,58	1,00	0,58
		Δ-Δ	0,58	1,00	0,58
		Y-Δ	0,33	0,88	0,88

Tableau 2-2 : Tensions de ligne au secondaire pour les creux de tension biphasés

	Amplitude	*Transformateur*	*Tensions de ligne au secondaire du transformateur (p.u)*		
			Vab	*Vbc*	*Vca*
Type C	10%	Y-Y	0,95	0,95	0,90
		Δ-Δ	0,95	0,95	0,90
		Y-Δ	0,92	0,96	0,92
	30%	Y-Y	0,86	0,86	0,70
		Δ-Δ	0,86	0,86	0,70
		Y-Δ	0,76	0,90	0,75
	60%	Y-Y	0,72	0,72	0,40
		Δ-Δ	0,72	0,72	0,40
		Y-Δ	0,53	0,80	0,53
	90%	Y-Y	0,61	0,61	0,18
		Δ-Δ	0,61	0,61	0,18
		Y-Δ	0,36	0,70	0,36
	100%	Y-Y	0,58	0,58	0,00
		Δ-Δ	0,58	0,58	0,00
		Y-Δ	0,33	0,67	0,33

Comme on pouvait s'y attendre, le type de transformateur utilisé affecte les tensions de ligne au secondaire. Les transformateurs Y-Y et Δ-Δ affectent les creux de tensions de la même façon. En présence de creux de tension monophasés, ces deux types de transformateurs affectent deux tensions de lignes tandis que le transformateur Y-Δ affecte toutes les tensions.

Pour les creux de tension monophasés, dans un cas extrême (creux de tension 100%) et quelque soit la configuration des transformateurs présentés dans cette étude, les tensions de ligne ne pourront aller plus bas que 33% de la valeur nominale.

Pour les creux de tension biphasés et dans un cas extrême, les tensions de ligne peuvent s'annuler pour les transformateurs Y-Y et Δ-Δ tandis que pour le transformateur Y-Δ, les tensions de lignes ne pourront aller plus bas que 33% de la valeur nominale. Ce dernier point peut être vu comme un avantage du

transformateur Y-Δ par rapport aux deux autres. Comme nous le verrons dans la suite de travail, le transformateur Y-Δ est utilisé dans le système auxiliaire de compensation de creux de tension que nous proposons.

2.4 Analyse d'un redresseur triphasé à diodes

Considérons le redresseur triphasé à diodes de la figure 2.5.

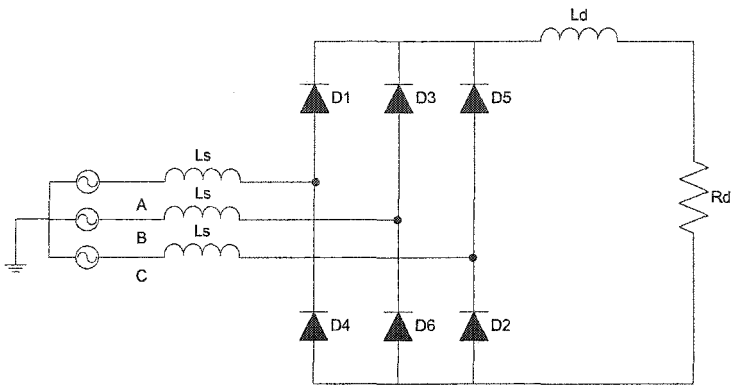


Figure 2.5 : Redresseur triphasé à diodes

Nous allons déterminer la valeur moyenne de la tension redressée à l'aide des fonctions de commutation. L'analyse de ce redresseur sera faite en régime équilibré et en régime déséquilibré.

Les hypothèses d'analyse sont :

- La source et les interrupteurs de puissance sont parfaits;
- Le redresseur fonctionne en mode de conduction continue;
- Les tensions sont sinusoïdales;
- Les inductances de lignes sont identiques.

2.4.1 Analyse du redresseur à diodes en régime équilibré

Il s'agit ici d'une analyse dans les conditions normales de fonctionnement donc en l'absence de creux de tension ou dans le cas d'un creux de type A. Nous utiliserons les fonctions de commutation que nous noterons S_a, S_b et S_c afin de déterminer la valeur moyenne de sortie d'un redresseur triphasé à diodes [13]. S_a, S_b et S_c sont des fonctions de commutation relatives aux états de conduction et de blocage des diodes de chaque phase d'alimentation du redresseur. Ces fonctions de commutation sont présentées à la figure 2.6.

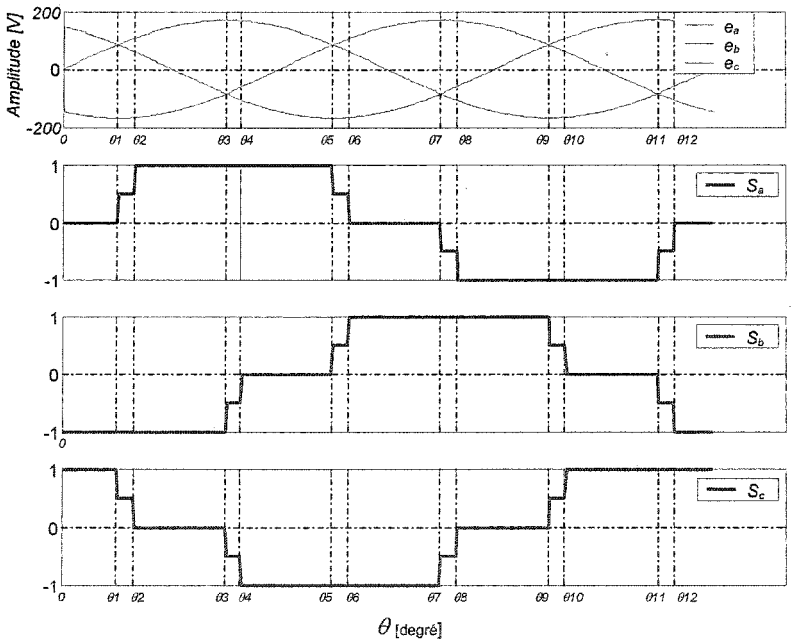


Figure 2.6 : Fonctions de commutation

Dans le cas d'une alimentation équilibrée, les instants de commutation sont définis comme suit :

$\theta_1 = \pi/6;$ $\theta_2 = \theta_1 + \mu$

$\theta_3 = \pi/2;$ $\theta_4 = \theta_3 + \mu$

$\theta_5 = 5\pi/6;$ $\theta_6 = \theta_5 + \mu$

$\theta_7 = 7\pi/6;$ $\theta_8 = \theta_7 + \mu$

$\theta_9 = 3\pi/2;$ $\theta_{10} = \theta_9 + \mu$

$\theta_{11} = 11\pi/6;$ $\theta_{12} = \theta_{11} + \mu$

μ est l'angle d'empiètement.

Les fonctions de commutation sont définies comme suit :

$$S_a = \begin{cases} -1 & si \quad \theta_8 \le \theta \le \theta_{11} \\ -0.5 & si \quad \theta_7 \le \theta \le \theta_8 \quad et \quad \theta_{11} \le \theta \le \theta_{12} \\ 0 & si \quad 0 \le \theta \le \theta_1 \quad et \quad \theta_6 \le \theta \le \theta_7 \quad et \quad \theta_{12} \le \theta \le 2\pi \\ 0.5 & si \quad \theta_1 \le \theta \le \theta_2 \quad et \quad \theta_5 \le \theta \le \theta_6 \\ 1 & si \quad \theta_2 \le \theta \le \theta_5 \end{cases}$$

$$S_b = \begin{cases} -1 & si \quad 0 \le \theta \le \theta_3 \quad et \quad \theta_{12} \le \theta \le 2\pi \\ -0.5 & si \quad \theta_3 \le \theta \le \theta_4 \quad et \quad \theta_{11} \le \theta \le \theta_{12} \\ 0 & si \quad \theta_4 \le \theta \le \theta_5 \quad et \quad \theta_{10} \le \theta \le \theta_{11} \\ 0.5 & si \quad \theta_5 \le \theta \le \theta_6 \quad et \quad \theta_9 \le \theta \le \theta_{10} \\ 1 & si \quad \theta_6 \le \theta \le \theta_9 \end{cases}$$

$$S_c = \begin{cases} -1 & si \quad \theta_4 \le \theta \le \theta_7 \\ -0.5 & si \quad \theta_3 \le \theta \le \theta_4 \quad et \quad \theta_7 \le \theta \le \theta_8 \\ 0 & si \quad \theta_2 \le \theta \le \theta_3 \quad et \quad \theta_8 \le \theta \le \theta_9 \\ 0.5 & si \quad \theta_1 \le \theta \le \theta_2 \quad et \quad \theta_9 \le \theta \le \theta_{10} \\ 1 & si \quad 0 \le \theta \le \theta_1 \quad et \quad \theta_{10} \le \theta \le 2\pi \end{cases}$$

La tension à la sortie du redresseur est définie comme suit [13] :

$$e_d = S_a e_a + S_b e_b + S_c e_c - (S_a^2 + S_b^2 + S_c^2) I_d R_s \qquad (2.8)$$

où e_a, e_b et e_c représentent la source de tension triphasée et sont donnés par :

$$e_a = V_a \sin\theta$$

$$e_b = V_b \sin\left(\theta - \frac{2\pi}{3}\right)$$

$$e_c = V_c \sin\left(\theta + \frac{2\pi}{3}\right)$$

Comme hypothèse d'analyse, nous considérons R_s, la résistance de l'impédance de la source, négligeable et (2.8) devient :

$$e_d = S_a e_a + S_b e_b + S_c e_c \tag{2.9}$$

Le développement en série de Fourier de la tension redressée est donné par :

$$e_d = U_{do} + \sum_{m=6}^{\infty} \left(A_{dm} \cos m\theta + B_{dm} \sin m\theta\right) \tag{2.10}$$

où U_{do} est la valeur moyenne de la tension e_d

et A_{dm}, B_{dm} sont les coefficients de Fourier des composantes harmoniques de la tension redressée e_d.

Dans cette analyse, nous nous intéressons seulement à la valeur moyenne de la tension redressée, alors :

$$U_{do} = \frac{1}{2\pi} \int_0^{2\pi} \left[S_a e_a + S_b e_b + S_c e_c\right] \, d\theta \tag{2.11}$$

L'équation (2.11) donne : $U_{do} = \frac{1}{2\pi}\left[\int_0^{2\pi}\left[S_a e_a\right]d\theta + \int_0^{2\pi}\left[S_b e_b\right]d\theta + \int_0^{2\pi}\left[S_c e_c\right]d\theta\right]$

Posons :

$$U_{doA} = \int_0^{2\pi}\left[S_a e_a\right]d\theta$$

$$U_{doB} = \int_0^{2\pi}\left[S_b e_b\right]d\theta$$

$$U_{doC} = \int_0^{2\pi}\left[S_c e_c\right]d\theta$$

Après développement, nous obtenons :

$$U_{doA} = \sqrt{3}V_a(1+\cos\mu)$$

$$U_{doB} = \sqrt{3}V_b(1+\cos\mu)$$

$$U_{doC} = \sqrt{3}V_c(1+\cos\mu)$$

donc, $U_{do} = \frac{1}{2\pi}[U_{doA} + U_{doB} + U_{doC}] = \frac{\sqrt{3}}{2\pi}(1+\cos\mu)[V_a + V_b + V_c]$

si $V_a = V_b = V_c = V_{LN}$, l'amplitude de la tension ligne-neutre, alors

$$U_{do} = \frac{3\sqrt{3}}{2\pi}V_{LN}(1+\cos\mu)$$

Ou encore,

$$U_{do} = \frac{3}{2\pi}V_{LL}(1+\cos\mu)$$

où V_{LL} est l'amplitude de la tension ligne-ligne.

En négligeant le phénomène d'empiètement, nous obtenons :

$$U_{do} = \frac{3\sqrt{3}}{\pi}V_{LN} = \frac{3}{\pi}V_{LL} \qquad (2.12)$$

Les mêmes expressions sont trouvées dans [17] pour la même analyse.

2.4.2 Analyse du redresseur à diodes en régime déséquilibré

En présence de creux de tension déséquilibrés, les intervalles des fonctions de commutation changent, car les bornes de ces intervalles représentent les passages par zéro des tensions de commutations (e_{ac}, e_{ba}, e_{cb}, e_{ca}, e_{ab} et e_{bc}).

Par exemple θ_1 et θ_7 seront trouvés pour $e_a = e_c$ (c-à-d $e_{ac} = 0$); en substituant e_a et e_c par leur expression on a :

$$\begin{cases} V_a \sin\theta_1 = V_c \sin\left(\theta_1 + \dfrac{2\pi}{3}\right) \\ V_a \sin\theta_7 = V_c \sin\left(\theta_7 + \dfrac{2\pi}{3}\right) \end{cases} \tag{2.13}$$

après développement trigonométrique, nous obtenons deux angles θ_1 et θ_7 :

$$\theta_1 = \tan^{-1}\left(\frac{\sqrt{3}V_c}{2V_a + V_c}\right) \tag{2.14}$$

$$\theta_7 = \tan^{-1}\left(\frac{\sqrt{3}V_c}{2V_a + V_c}\right) + \pi \tag{2.15}$$

θ_3 et θ_9 seront trouvés pour $e_c = e_b$ (c-à-d $e_{bc} = 0$) et nous obtenons :

$$\theta_3 = \tan^{-1}\left(\frac{\sqrt{3}(V_c + V_b)}{(V_c - V_b)}\right) \tag{2.16}$$

$$\theta_9 = \tan^{-1}\left(\frac{\sqrt{3}(V_c + V_b)}{(V_c - V_b)}\right) + \pi \tag{2.17}$$

θ_5 et θ_{11} seront trouvés pour $e_a = e_b$ (c-à-d $e_{ba} = 0$) et nous obtenons :

$$\theta_5 = \tan^{-1}\left(\frac{-\sqrt{3}V_b}{2V_a + V_b}\right) \tag{2.18}$$

$$\theta_{11} = \tan^{-1}\left(\frac{-\sqrt{3}V_b}{2V_a + V_b}\right) + \pi \tag{2.19}$$

À partir des expressions (2.14) à (2.19), on peut voir que les instants de commutation des fonctions S_a, S_b et S_c dépendent essentiellement de l'amplitude des tensions phase-neutre.

On a :

$$U_{do} = \frac{1}{2\pi}\left[\int_0^{2\pi}[S_a e_a]\,d\theta + \int_0^{2\pi}[S_b e_b]\,d\theta + \int_0^{2\pi}[S_c e_c]\,d\theta\right]$$

Posons :

$$U_{doA} = \int_0^{2\pi} [S_a e_a] d\theta$$

$$U_{doB} = \int_0^{2\pi} [S_b e_b] d\theta$$

$$U_{doC} = \int_0^{2\pi} [S_c e_c] d\theta$$

Après développement, nous obtenons :

$$U_{doA} = \frac{1}{2} V_a \{(1+\cos\mu)(\cos\theta_1 - \cos\theta_5 - \cos\theta_7 + \cos\theta_{11}) - \sin\mu\,(\sin\theta_1 - \sin\theta_5 - \sin\theta_7 + \sin\theta_{11})\}$$

$$U_{doB} = -\frac{1}{4} V_b \{(1+\cos\mu-\sqrt{3}\sin\mu)(\cos\theta_3 + \cos\theta_5 - \cos\theta_9 - \cos\theta_{11})\ldots$$

$$-(\sqrt{3}(1+\cos\mu)+\sin\mu)(\sin\theta_3 + \sin\theta_5 - \sin\theta_9 - \sin\theta_{11})\}$$

$$U_{doC} = \frac{1}{4} V_c \{(1+\cos\mu+\sqrt{3}\sin\mu)(\cos\theta_1 + \cos\theta_3 - \cos\theta_7 - \cos\theta_9)\ldots$$

$$+(\sqrt{3}(1+\cos\mu)-\sin\mu)(\sin\theta_1 + \sin\theta_3 - \sin\theta_7 - \sin\theta_9)\}$$

En négligeant l'effet d'empiètement sur la tension moyenne, nous obtenons :

$$U_{doA} = V_a \{\cos\theta_1 - \cos\theta_5 - \cos\theta_7 + \cos\theta_{11}\} \quad (2.20)$$

$$U_{doB} = -\frac{1}{2} V_b \{(\cos\theta_3 + \cos\theta_5 - \cos\theta_9 - \cos\theta_{11}) - \sqrt{3}\,(\sin\theta_3 + \sin\theta_5 - \sin\theta_9 - \sin\theta_{11})\} \quad (2.21)$$

$$U_{doC} = \frac{1}{2} V_c \{(\cos\theta_1 + \cos\theta_3 - \cos\theta_7 - \cos\theta_9) + \sqrt{3}\,(\sin\theta_1 + \sin\theta_3 - \sin\theta_7 - \sin\theta_9)\} \quad (2.22)$$

avec $U_{do} = \frac{1}{2\pi}[U_{doA} + U_{doB} + U_{doC}]$

Les résultats de calcul à l'aide des équations et les résultats de simulation obtenus à l'aide de PSB (Power System Blockset, maintenant SimPowerSystems) et de PSIM

[34] sont représentés dans le tableau 2-3 et sur la figure 2.7. La précision des résultats obtenus avec l'approche basée sur les fonctions de commutation est donc vérifiée, les faibles écarts obtenus avec PSB étant dus à la différence des modèles des interrupteurs. Par exemple, notre analyse considère des interrupteurs parfaits c'est-à-dire une résistance nulle pour un interrupteur fermé et une résistance infinie pour un interrupteur ouvert tandis que PSIM considère une résistance élevée de 1MΩ pour un interrupteur ouvert et une résistance faible 10μΩ pour un interrupteur fermé [34].

Tableau 2-3 : Valeurs moyennes de la tension redressée pour différentes conditions de la tension d'alimentation

		Valeur moyenne [V]		
Creux de type	***(Va,Vb,Vc) [V]***	***PSB***	***PSIM***	***Approche analytique***
équilibré	*(120, 120, 120)*	279,1	280,76	280,69
	(80, 80, 80)	185,53	187,24	187,13
déséquilibré	*(20, 120, 120)*	210,04	211,73	211,64
	(120, 60, 120)	234,88	236,58	236,48
	(120, 120, 80)	248,94	250,65	250,54
	(20, 20, 120)	132,07	133,80	133,67
	(120, 20, 12)	126,95	128,69	128,55
	(15, 120, 12)	123,55	125,30	125,15

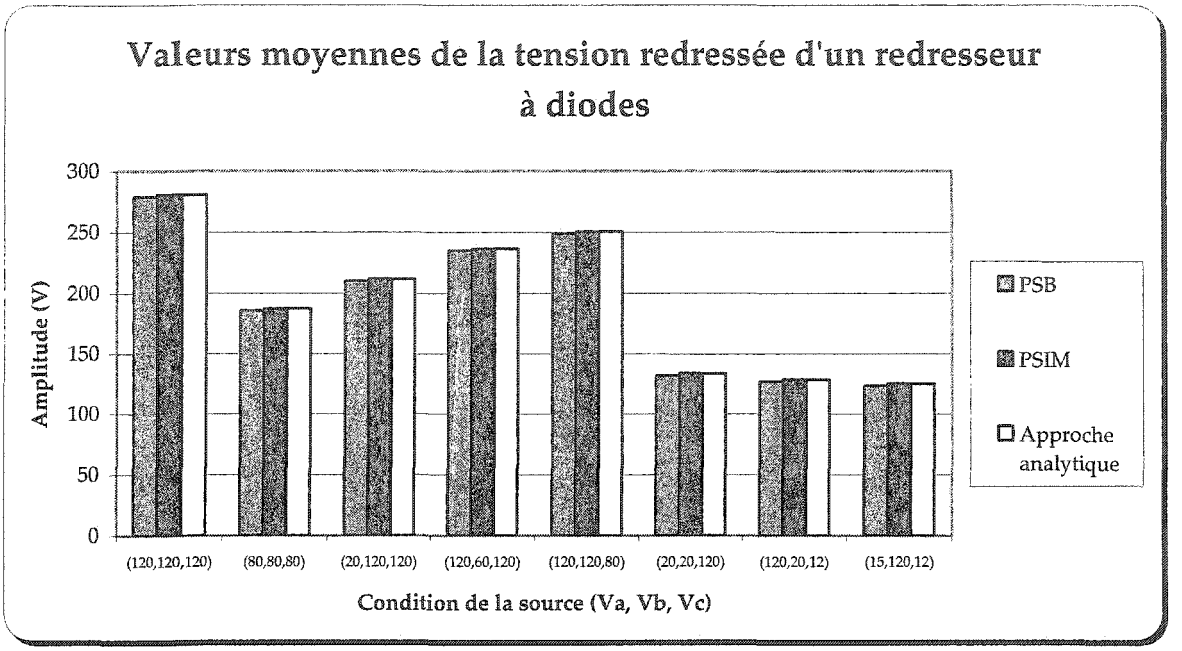


Figure 2.7 : Valeurs moyennes de la tension redressée en fonction de différentes conditions de la tension d'alimentation

2.5 Analyse d'un redresseur triphasé à thyristors

Considérons le redresseur triphasé à thyristors de la figure 2.8 :

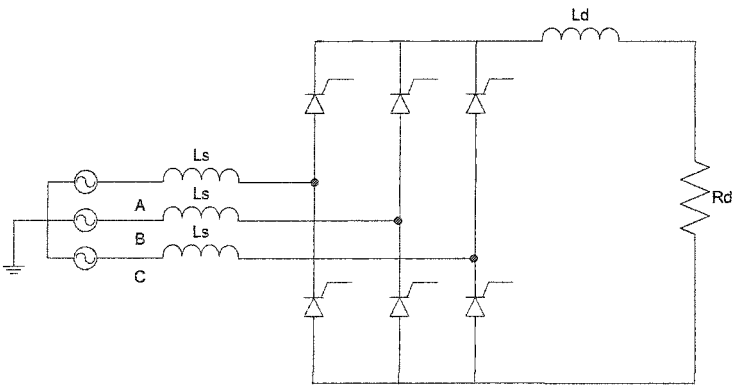


Figure 2.8 : Redresseur triphasé à thyristors

Comme dans le cas du redresseur à diodes, nous allons déterminer la valeur moyenne de la tension redressée à l'aide des fonctions de commutation. L'analyse de ce redresseur sera faite en régime équilibré et en régime déséquilibré.

2.5.1 Analyse du redresseur à thyristors en régime équilibré

Nous utilisons le principe des fonctions de commutation exposé à la section précédente. Les fonctions de commutation ont une forme identique à celles présentées dans le cas d'un redresseur à diodes. Elles subissent une translation positive d'un angle α.

Dans le cas d'une alimentation triphasée équilibrée, les instants de commutation pour le redresseur à diodes sont définis comme suit :

$\theta_1 = \pi/6;$ $\theta_2 = \theta_1 + \mu$

$\theta_3 = \pi/2;$ $\theta_4 = \theta_3 + \mu$

$\theta_5 = 5\pi/6;$ $\theta_6 = \theta_5 + \mu$

$\theta_7 = 7\pi/6;$ $\theta_8 = \theta_7 + \mu$

$\theta_9 = 3\pi/2;$ $\theta_{10} = \theta_9 + \mu$

$\theta_{11} = 11\pi/6;$ $\theta_{12} = \theta_{11} + \mu$

après translation positive de l'angle α, on a :

$\phi_0 = \alpha;$

$\phi_1 = \theta_1 + \alpha;$ $\phi_2 = \phi_1 + \mu$

$\phi_3 = \theta_3 + \alpha;$ $\phi_4 = \phi_3 + \mu$

$\phi_5 = \theta_5 + \alpha;$ $\phi_6 = \phi_5 + \mu$

$\phi_7 = \theta_7 + \alpha;$ $\phi_8 = \phi_7 + \mu$

$\phi_9 = \theta_9 + \alpha;$ $\phi_{10} = \phi_9 + \mu$

$\phi_{11} = \theta_{11} + \alpha;$ $\phi_{12} = \phi_{11} + \mu$

$\phi_{13} = 2\pi + \alpha$

μ est l'angle d'empiètement.

L'équation (2.11) donne : $U_{do\alpha} = \frac{1}{2\pi}\left[\int_{\phi_0}^{\phi_{13}}[S_a e_a]\,d\phi + \int_{\phi_0}^{\phi_{13}}[S_b e_b]\,d\phi + \int_{\phi_0}^{\phi_{13}}[S_c e_c]\,d\phi\right]$

Posons :

$$U_{do\alpha A} = \int_{\phi_0}^{\phi_{13}}[S_a e_a]\,d\phi$$

$$U_{do\alpha B} = \int_{\phi_0}^{\phi_{13}}[S_b e_b]\,d\phi$$

$$U_{do\alpha C} = \int_{\phi_0}^{\phi_{13}}[S_c e_c]\,d\phi$$

Après développement, nous obtenons :

$$U_{do\alpha A} = \sqrt{3}\,V_a\{(1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha\}$$

$$U_{do\alpha B} = \sqrt{3}\,V_b\{(1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha\}$$

$$U_{do\alpha C} = \sqrt{3}\,V_c\{(1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha\}$$

donc,

$$U_{do\alpha} = \frac{1}{2\pi}[U_{do\alpha A} + U_{do\alpha B} + U_{do\alpha C}] = \frac{\sqrt{3}}{2\pi}((1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha)[V_a + V_b + V_c]$$

Si $V_a = V_b = V_c = V_{LN}$, l'amplitude de la tension ligne-neutre, alors

$$U_{do\alpha} = \frac{3\sqrt{3}}{2\pi}V_{LN}\,((1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha) \qquad (2.23)$$

Ou encore,

$$U_{do\alpha} = \frac{3}{2\pi}V_{LL}\,((1+\cos\mu)\cos\alpha - \sin\mu\sin\alpha) \qquad (2.24)$$

où V_{LL} est l'amplitude de la tension ligne-ligne.

En négligeant le phénomène d'empiètement, nous obtenons :

$$U_{do\alpha} = \frac{3\sqrt{3}}{\pi}V_{LN}\cos\alpha = \frac{3}{\pi}V_{LL}\cos\alpha = U_{do}\cos\alpha \qquad (2.25)$$

où U_{do} est la valeur moyenne d'un redresseur triphasé à diodes.

Les mêmes expressions sont trouvées en [17] pour la même analyse.

2.5.2 Analyse du redresseur à thyristors en régime déséquilibré

Comme dans le cas d'un convertisseur à diodes, il est important de recalculer les instants de commutation des fonctions S_a, S_b et S_c car ceux-ci dépendent des amplitudes des tensions ligne-neutre de la source comme le démontrent les équations (2.14) à (2.19).

L'équation (2.11) donne : $U_{do\alpha} = \frac{1}{2\pi}\left[\int_{\phi_0}^{\phi_{13}}[S_a e_a]\,d\phi + \int_{\phi_0}^{\phi_{13}}[S_b e_b]\,d\phi + \int_{\phi_0}^{\phi_{13}}[S_c e_c]\,d\phi\right]$

Posons :

$$U_{do\alpha A} = \int_{\phi_0}^{\phi_{13}}[S_a e_a]\,d\phi$$

$$U_{do\alpha B} = \int_{\phi_0}^{\phi_{13}}[S_b e_b]\,d\phi$$

$$U_{do\alpha C} = \int_{\phi_0}^{\phi_{13}}[S_c e_c]\,d\phi$$

Après développement, nous obtenons :

$$U_{do\alpha A} = \frac{1}{2}V_a\{(1+\cos\mu)(\cos\phi_1 - \cos\phi_5 - \cos\phi_7 + \cos\phi_{11}) - \sin\mu\,(\sin\phi_1 - \sin\phi_5 - \sin\phi_7 + \sin\phi_{11})\}$$

$$U_{do\alpha B} = -\frac{1}{4}V_b\{(1+\cos\mu-\sqrt{3}\sin\mu)(\cos\phi_3 + \cos\phi_5 - \cos\phi_9 - \cos\phi_{11})\ldots$$
$$-(\sqrt{3}(1+\cos\mu)+\sin\mu)(\sin\phi_3 + \sin\phi_5 - \sin\phi_9 - \sin\phi_{11})\}$$

$$U_{do\alpha C} = \frac{1}{4}V_c\{(1+\cos\mu+\sqrt{3}\sin\mu)(\cos\phi_1 + \cos\phi_3 - \cos\phi_7 - \cos\phi_9)\ldots$$
$$+(\sqrt{3}(1+\cos\mu)-\sin\mu)(\sin\phi_1 + \sin\phi_3 - \sin\phi_7 - \sin\phi_9)\}$$

En négligeant l'effet d'empiètement sur la tension moyenne, nous obtenons :

$$U_{do\alpha A} = V_a\{(\cos\phi_1 - \cos\phi_5 - \cos\phi_7 + \cos\phi_{11})\} \qquad (2.26)$$

$$U_{do\alpha B} = -\frac{1}{2}V_b\{\ (\cos\phi_3 + \cos\phi_5 - \cos\phi_9 - \cos\phi_{11})... \qquad (2.27)$$

$$-\sqrt{3}\,(\sin\phi_3 + \sin\phi_5 - \sin\phi_9 - \sin\phi_{11})\}$$

$$U_{do\alpha C} = \frac{1}{2}V_c\{\ (\cos\phi_1 + \cos\phi_3 - \cos\phi_7 - \cos\phi_9)... \qquad (2.28)$$

$$+\sqrt{3}\,(\sin\phi_1 + \sin\phi_3 - \sin\phi_7 - \sin\phi_9)\ \}$$

avec $U_{do\alpha} = \frac{1}{2\pi}\left[U_{do\alpha A} + U_{do\beta B} + U_{do\alpha C}\right]$

2.6 Analyse d'un redresseur triphasé à thyristors avec diode de roue libre

Ce convertisseur tient sa particularité à l'ajout d'une diode dite diode de roue libre à la sortie d'un convertisseur à thyristors classique. Cette diode a pour rôle principal d'empêcher la valeur moyenne de la tension redressée de devenir négative. Ce convertisseur est présenté à la figure 2.9.

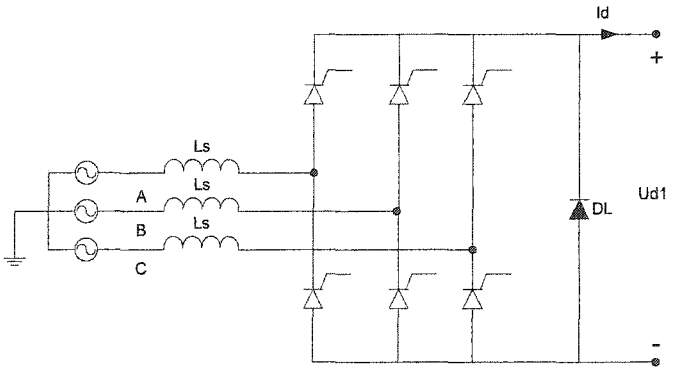


Figure 2.9 : Redresseur triphasé à thyristors avec diode de roue libre

Tant que la conduction des thyristors rend la tension U_{d1} positive, cette diode n'intervient pas car elle est polarisée en inverse. Lorsque U_{d1} tend à être négative, la diode D_L est polarisée en direct et devient passante. Le courant I_d ne pouvant s'annuler instantanément à cause de l'inductance de lissage L_d, il s'annulera en circulant dans la diode D_L.

Cette diode permet de réduire l'ondulation de la tension redressée pour les angles d'amorçage supérieurs à $\frac{\pi}{3}$; ceci veut dire que l'ajout de la diode de roue libre contribue à [18]:

- Réduire les harmoniques côté continu,
- Réduire les harmoniques des courants de lignes,
- Réduire le courant efficace dans les thyristors.

Les hypothèses d'analyse sont :

- La source et les interrupteurs de puissance sont parfaits,
- Mode de conduction continue tel que le courant I_d est considéré constant.

Le redresseur triphasé à thyristors équivaut à une source de tension V_s de période $\frac{\pi}{3}$ en série avec une diode D. La figure 2.10 représente le schéma équivalent du redresseur triphasé à thyristors et de la diode de roue libre.

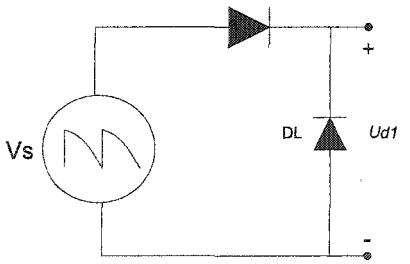


Figure 2.10 : Schéma équivalent d'un redresseur triphasé à thyristors avec diode de roue libre

V_s a pour expression : $V_s = V_{LL}\cos\theta$, pour $\quad -\frac{\pi}{6}+\alpha < \theta < \frac{\pi}{6}+\alpha$

où V_{LL} est la tension ligne-ligne.

Pour $\alpha < \frac{\pi}{3}$, V_s demeure positive et la diode D_L ne conduit pas. Donc la tension moyenne est celle de d'un redresseur à thyristors sans diode de roue libre, soit :

$$U_{d1o} = U_{do}\cos\alpha$$

où U_{do} est la valeur moyenne d'un redresseur triphasé à diodes.

Pour l'intervalle où D_L intervient, donc pour $\alpha > \frac{\pi}{3}$, la valeur moyenne de sortie est :

$$U_{d1o} = \frac{6}{2\pi} \int_{-\frac{\pi}{6}+\alpha}^{\frac{\pi}{2}} V_{LL} \cos\theta \quad d\theta \tag{2.29}$$

L'équation (2.29) nous donne: $U_{d1o} = \frac{6V_{LL}}{2\pi}\left[\sin\frac{\pi}{2} - \sin\left(-\frac{\pi}{6}+\alpha\right)\right]$

$$U_{d1o} = \frac{3V_{LL}}{\pi}\left[1+\sin\left(\frac{\pi}{6}-\alpha\right)\right] \tag{2.30}$$

ou encore $$U_{d1o} = U_{do}\left[1+\sin\left(\frac{\pi}{6}-\alpha\right)\right] \tag{2.31}$$

U_{do} est la tension moyenne d'un redresseur à diodes.

2.7 Conclusion

Nous avons présenté de façon exhaustive les différents types de creux de tension selon [3]. Les effets des transformateurs sur le type de creux de tension ont été étudiés. Les résultats de simulation obtenus et présentés dans les tableaux 2-1 et 2-2 concordent avec ceux présentés par Mansoor et *al.* [15].

Le concept des fonctions de commutation nous a permis d'analyser et de déterminer des équations pour le calcul de la valeur moyenne de la tension redressée, en présence de creux de tension équilibrés et déséquilibrés pour les différents convertisseurs. Quelques applications numériques présentées dans le tableau II-3 témoignent de la validité des équations développées. Ces modèles sont très utiles pour l'élaboration d'une loi de compensation fiable.

CHAPITRE 3 : MÉTHODES DE DÉTECTION ET D'ESTIMATION DES CREUX DE TENSION

3.1 Introduction

Plusieurs techniques ont été proposées pour la détection des problèmes reliés à la qualité de l'alimentation électrique. Parmi celles-là, figure Adaline (***ada***ptive ***lin***ear ***ne***uron). C'est une approche qui utilise un réseau de neurones adaptatif ayant N entrées et une sortie. La sortie est une combinaison linéaire des *N* entrées. Adaline a été premièrement utilisé comme une méthode adaptative pour l'estimation de l'amplitude et de la phase du fondamental et des harmoniques d'un signal déformé [19]. Concernant l'estimation des harmoniques, une autre méthode a été beaucoup utilisée : *les ondelettes*. La transformée en ondelettes a été utilisée avec succès dans l'estimation des grandeurs efficaces de la tension, du courant et de la puissance [4]. Dans la détection des problèmes reliés à la qualité de l'alimentation, selon [20] cette méthode demeure cependant tributaire de l'ondelette mère choisie. Adaline par contre représente une approche intéressante pour l'estimation des harmoniques [20]. La simplicité de cette méthode est liée à ses calculs simples (essentiellement des additions, soustractions et des multiplications) facilite son implémentation. Adaline est rapide grâce à sa construction simple. Ceci est un argument de taille lorsque vient le temps de choisir une méthode viable et surtout rapide pour la détection des creux de tension.

3.2 Application de *Adaline* à la détection des creux de tension

La problématique de la détection des creux de tension réside sur l'estimation de l'amplitude de la tension. Adaline ayant été utilisée dans l'estimation de l'amplitude et de la phase du fondamental et des harmoniques d'un signal, elle devient une avenue intéressante à explorer. Dans le cas des creux de tension, seules l'amplitude et la phase du fondamental nous intéressent. Les sous-sections qui suivent présentent le principe de la méthode Adaline, son algorithmique ainsi que les performances de cette méthode appliquée à la détection rapide de creux de tension.

3.2.1 Principe de la méthode Adaline

D'après l'analyse de Fourier, un signal périodique peut être décomposé en une somme de sinus et de cosinus. Le modèle du signal estimé est le suivant :

$$y(t) = \sum_{n=1}^{N} \left(X_n \cos(n\omega t) + Y_n \sin(n\omega t) \right) \qquad (3.1)$$

où X_n et Y_n sont les coefficients de la série de Fourier du signal $y(t)$; n est le rang harmonique; X_1 et Y_1 sont les coefficients de Fourier du fondamental.

Dans l'équation (3.1), la composante continue n'est pas représentée car le signal qui fait l'objet d'analyse dans ce travail est alternatif et sans composante continue.

Sous forme matricielle, l'équation (3.1) devient :

$$y(t) = W^T \cdot X(t) \qquad (3.2)$$

où

$$X(t) = \left[\cos\omega t \quad \sin\omega t \quad ..... \quad \cos N\omega t \quad \sin N\omega t\right]^T \text{ et } W = \begin{bmatrix} X_1 \\ Y_1 \\ . \\ . \\ . \\ X_N \\ Y_N \end{bmatrix} \qquad (3.3)$$

W est la matrice des poids qui doit être mise à jour à chaque nouvel échantillon du signal $y(t)$. La structure de la méthode Adaline est présentée à la figure 3.1.

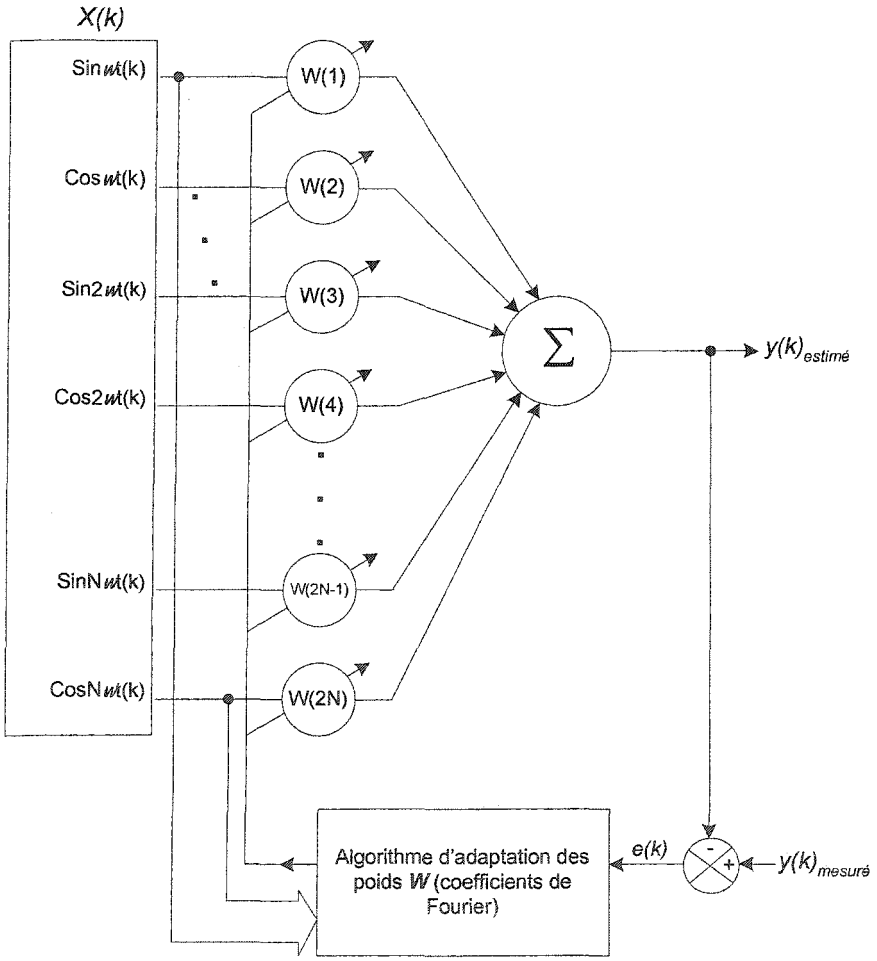


Figure 3.1 : Structure de la méthode Adaline

L'amplitude et la phase du fondamental se définissent de la manière suivante :

$$A_1 = \sqrt{X_1^2 + Y_1^2} = \sqrt{W(1) \times W(1) + W(2) \times W(2)} \qquad (3.4)$$

et

$$\phi_1 = a\tan\left(\frac{Y_1}{X_1}\right) = a\tan\left(\frac{W(2)}{W(1)}\right) \tag{3.5}$$

Pour l'algorithme d'adaptation des poids, notre choix s'est porté sur l'algorithme des moindres carrés récursifs (RLS). Il s'agit d'une méthode quadratique qui consiste à minimiser une fonction quadratique de l'erreur *e(k)* entre le signal mesuré et le modèle. C'est l'un des algorithmes les plus connus; il est utilisé dans le filtrage adaptatif et l'identification des systèmes. Il doit sa popularité principalement à sa grande vitesse de convergence.

La matrice des poids *W* (ou la matrice des coefficients de Fourier) est mise à jour en utilisant l'équation (3.6) :

$$W(k) = W(k-1) + K(k) \times e(k) \tag{3.6}$$

$$\text{avec} \quad K(k) = \frac{P(k-1)X(k)}{\lambda + X^T(k)P(k-1)X(k)} \tag{3.7}$$

$$\text{et} \quad P(k) = \left(\frac{1}{\lambda}\right)\left[I - K(k)X^T(k)\right]P(k-1) \tag{3.8}$$

K est la matrice des gains d'adaptation des coefficients de Fourier;

I est la matrice identité;

P correspond à l'inverse de la matrice d'autocorrelation du vecteur d'entrée X;

λ est le facteur d'oubli; ce facteur d'oubli permet de négliger progressivement les échantillons plus anciens. Cette variante du RLS permet l'identification de paramètres variants lentement dans le temps. La convergence est beaucoup plus rapide par rapport à un autre algorithme connu tel que LMS (Least Mean Square).

3.2.2 Algorithmique de la méthode Adaline

Dans la section précédente nous avons exposé la théorie de la méthode Adaline appliquée à la détection des creux de tension. Dans le cadre de ce travail, il ne s'agit pas seulement d'estimer l'amplitude d'un signal (la tension dans notre cas) de façon précise. Une détection (par une estimation) rapide est requise. Cette rapidité conditionnera la vitesse de la commande.

L'organigramme de la méthode Adaline est présenté à la figure 3.1. Cet algorithme donne des résultats avec des précisions acceptables en régime stationnaire. En régime transitoire, c'est-à-dire à l'apparition d'un creux de tension, cet algorithme demeure relativement lent.

Pour améliorer les performances (rapidité et précision) de cet algorithme en présence de creux de tension, nous proposons une légère modification à l'algorithme Adaline de base tel qu'il est présenté à la figure 3.2. Il s'agit de la ***supervision de l'erreur entre le signal mesuré et le signal estimé.*** Cette supervision consiste à vérifier à chaque fois l'erreur calculée. Si elle devient plus grande qu'un seuil fixé au préalable, une réinitialisation de l'inverse de la matrice d'autocorrelation P est effectuée. L'organigramme de la méthode Adaline modifiée est présenté à la figure 3.3.

Le code Matlab de ces algorithmes est présenté à l'annexe A.

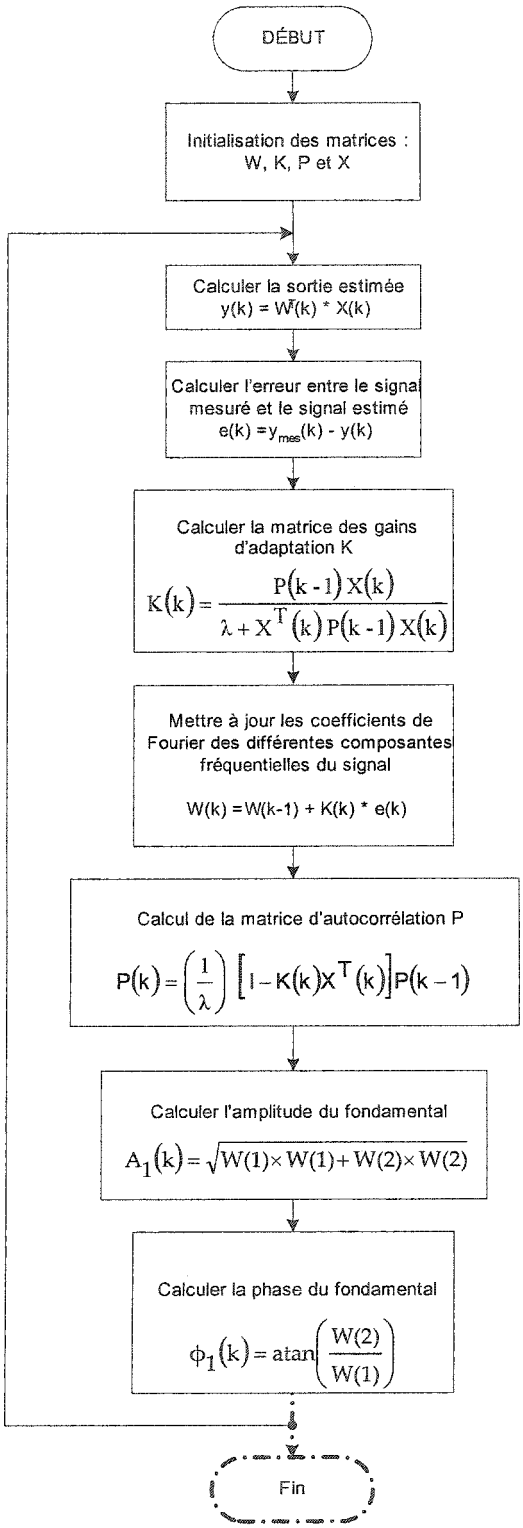


Figure 3.2 : Organigramme de la méthode Adaline

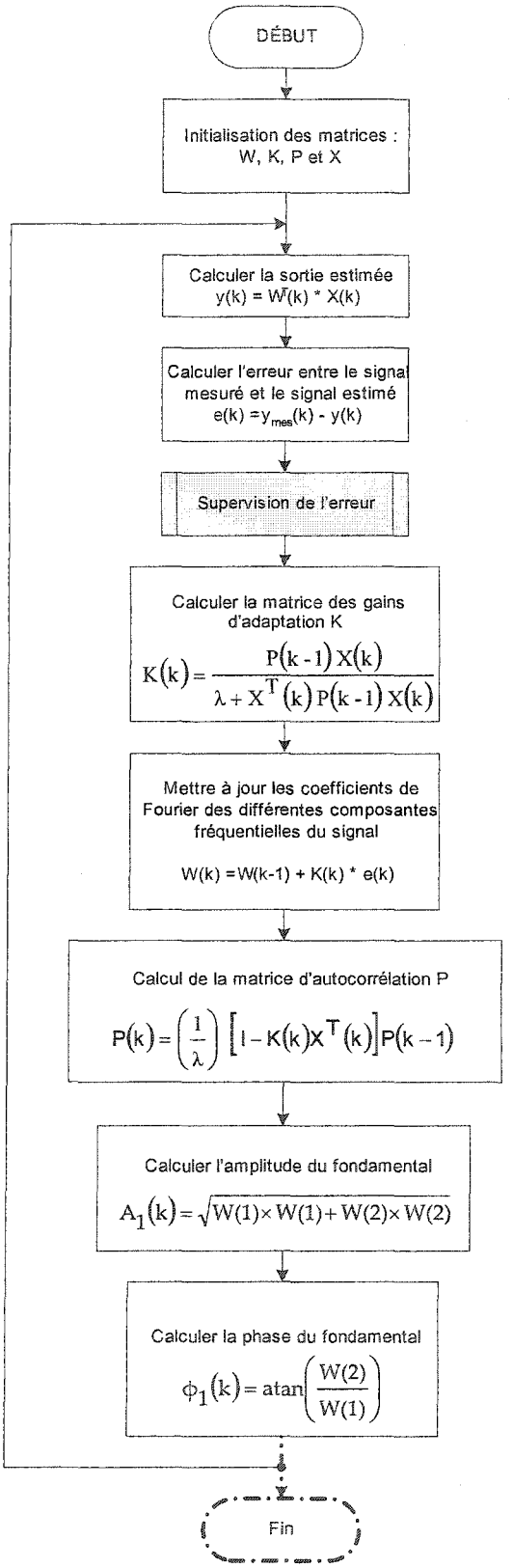


Figure 3.3 : Organigramme de la méthode Adaline modifiée

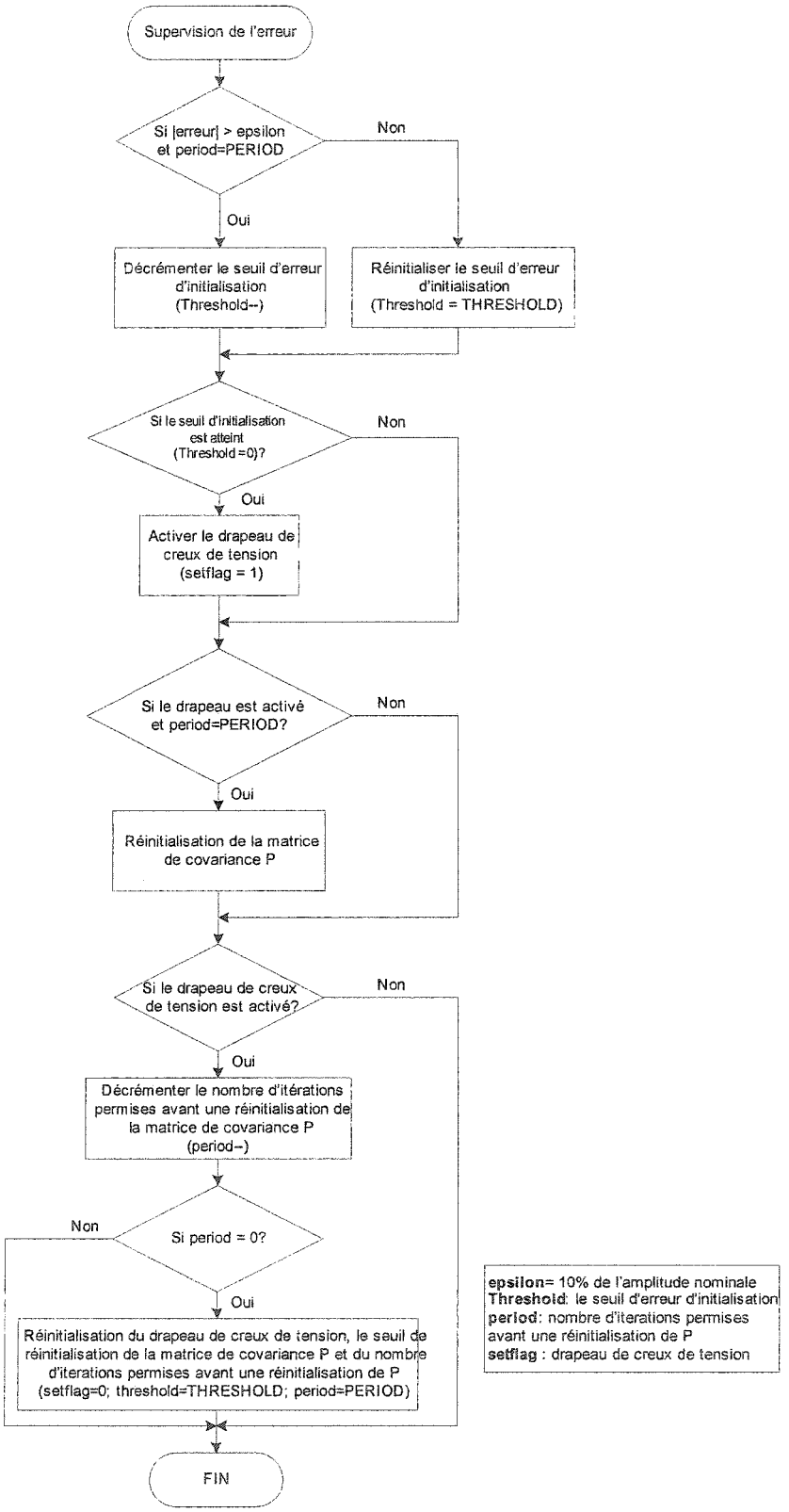


Figure 3.4 : Organigramme de la supevision de l'erreur

3.3 Performances de la méthode Adaline dans la détection rapide des creux de tension

Cette section présente les performances des deux méthodes Adaline (sans supervision et avec supervision de l'erreur) appliquées à la détection et l'estimation des creux de tension.

Les simulations faites avec Matlab considèrent une tension bruitée (signal synthétique) dans le lequel on retrouve le fondamental et les harmoniques de rang 3, 5, 7 et 9. Ce signal se définit comme suit :

$$y(t) = 220\sin(\omega t + 80°) + 11\sin(3\omega t + 60°)$$
$$+ 5.5\sin(5\omega t + 45°) + 2.64\sin(7\omega t + 36°) + 1.32\sin(9\omega t + 30°) + b(t)$$

$b(t)$ est un bruit aléatoire dont la valeur maximale correspond à 0.5% de la valeur maximale du fondamental ;

$\omega = 2\pi f$ est la pulsation pour une fréquence de 60Hz;

la fréquence d'échantillonnage est 15360 éch./s soit 256 échantillons par cycle ;

le facteur d'oubli $\lambda = 0.96$;

les matrices W, K, X et P s'initialisent de la façon suivante :

$W = [0 \quad 0 \quad 0 \quad .... \quad 0]^T$; dim(W) = 2N×1

$K = [0 \quad 0 \quad 0 \quad .... \quad 0]^T$; dim(K) = 2N×1

$X = [\sin\omega t \quad \cos\omega t \quad ..... \quad \sin N\omega t \quad \cos N\omega t]_{t=kT_s}$; dim(X) = 1×2N

$$P = \begin{bmatrix} 120 & 0 & ... & 0 \\ 0 & 120 & ... & 0 \\ . & . & . & . \\ 0 & 0 & 0 & 120 \end{bmatrix} ; \dim(P) = 2N \times 2N$$

N : nombre d'harmoniques considérés ;

T_s : période d'échantillonnage.

Les tableaux 3-1 et 3-2 nous permettent de vérifier la précision de la méthode Adaline en régime stationnaire.

Tableau 3-1 : Estimation des amplitudes par la méthode Adaline

	Algorithme sans Supervision de l'erreur			*Algorithme avec supervision de l'erreur*		
	Amplitudes			*Amplitudes*		
Rang harmonique	*Réelles(V)*	*Estimées (V)*	*Erreur relative (%)*	*Réelles (V)*	*Estimés (V)*	*Erreur relative (%)*
fondamental	220.00	220.42	0.19	220.00	219.85	0.07
3	11.00	11.58	5.29	11.00	11.00	0.01
5	5.50	5.57	1.20	5.50	5.65	2.68
7	2.64	2.81	6.61	2.64	2.68	1.42
9	1.32	1.25	5.55	1.32	1.31	0.49

Tableau 3-2 : Estimation des phases par la méthode Adaline

	Algorithme sans supervision de l'erreur			*Algorithme avec supervision de l'erreur*		
	Phases			*Phases*		
Rang harmonique	*Réelles (°)*	*Estimées (°)*	*Erreur relative (%)*	*Réelles (°)*	*Estimées (°)*	*Erreur relative (%)*
fondamental	80.00	79.93	0.09	80.00	80.02	0.02
3	60.00	59.86	0.24	60.00	59.66	0.57
5	45.00	42.93	4.60	45.00	43.97	2.29
7	36.00	31.10	13.61	36.00	32.75	9.03
9	30.00	33.25	10.82	30.00	30.80	2.66

L'estimation de l'amplitude du fondamental est l'opération qui offre la meilleure précision. Parlant toujours du fondamental, nous constatons que la supervision de l'erreur contribue à améliorer la précision de l'estimateur même en régime stationnaire (voir Tableau 3-1 et 3-2).

Les figures 3.5, 3.6 et 3.7 nous amènent à la conclusion que la supervision de l'erreur améliore la rapidité de l'algorithme pour la détection d'un creux de tension.

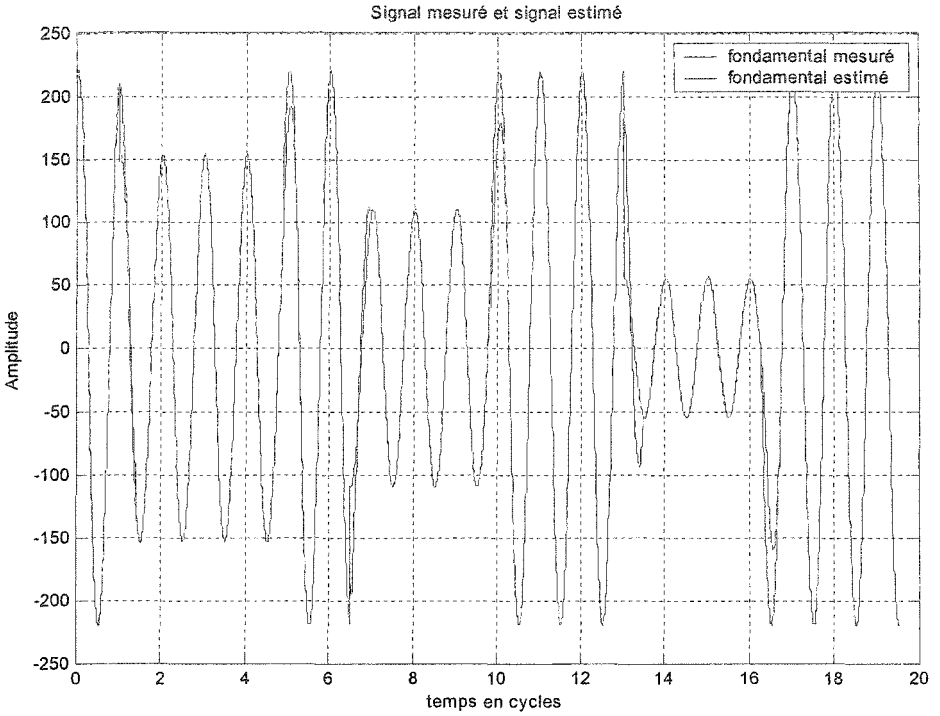


Figure 3.5 : Performances de la méthode Adaline sans supervision de l'erreur

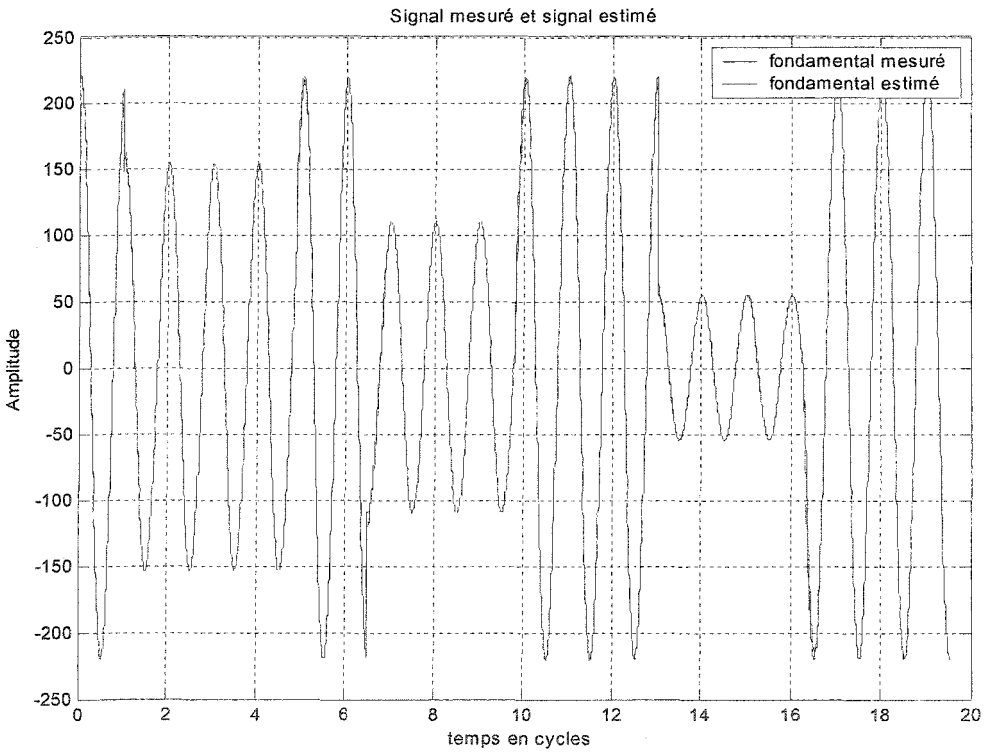


Figure 3.6 : Performances de la méthode Adaline avec supervision de l'erreur

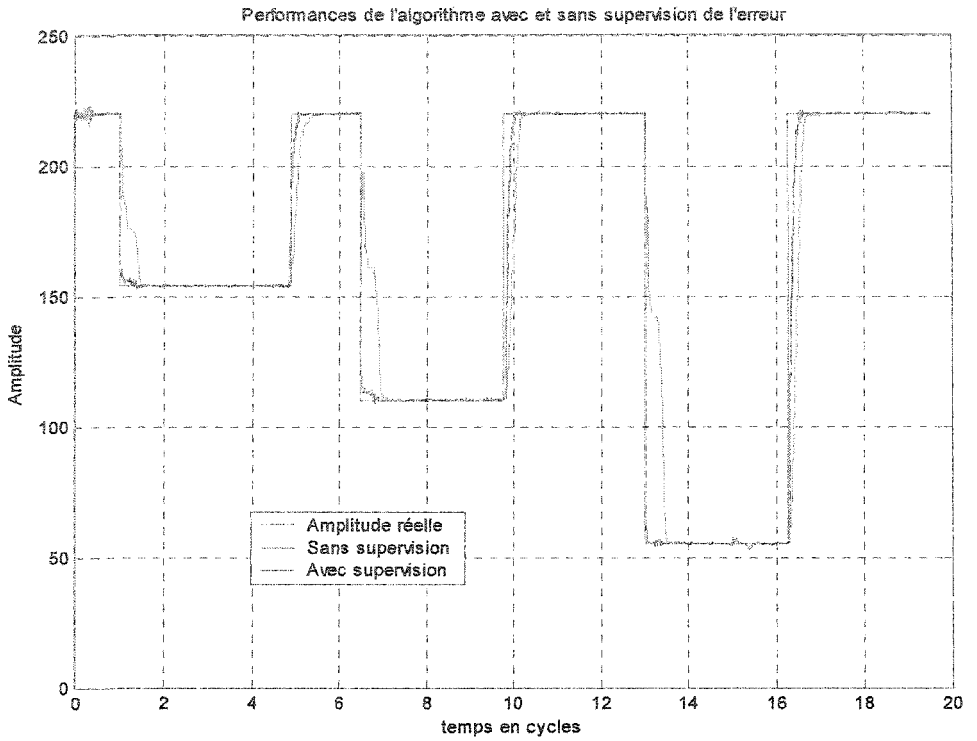


Figure 3.7 : Estimation de l'amplitude en présence de creux de tension

Afin de vérifier la rapidité de nos deux algorithmes, nous avons fait des agrandissements (voir figures 3.8 et 3.9) et évalué le délai lors de la détection dans les régions critiques; à savoir au début et à la fin d'un creux de tension. Les valeurs obtenues sont présentées dans le tableau 3-3. On peut noter que la rapidité n'est pas la même au début du creux et à la fin du creux. La supervision de l'erreur a largement contribué à améliorer la rapidité de l'algorithme Adaline.

Tableau 3-3 : Délai lors de la détection de creux de tension

	Délai lors de la détection	
Région critique	***Algorithme sans supervision de l'erreur***	***Algorithme avec supervision de l'erreur***
Début du creux	8 ms	0.83 ms
Fin du creux	10.6 ms	4.8 ms

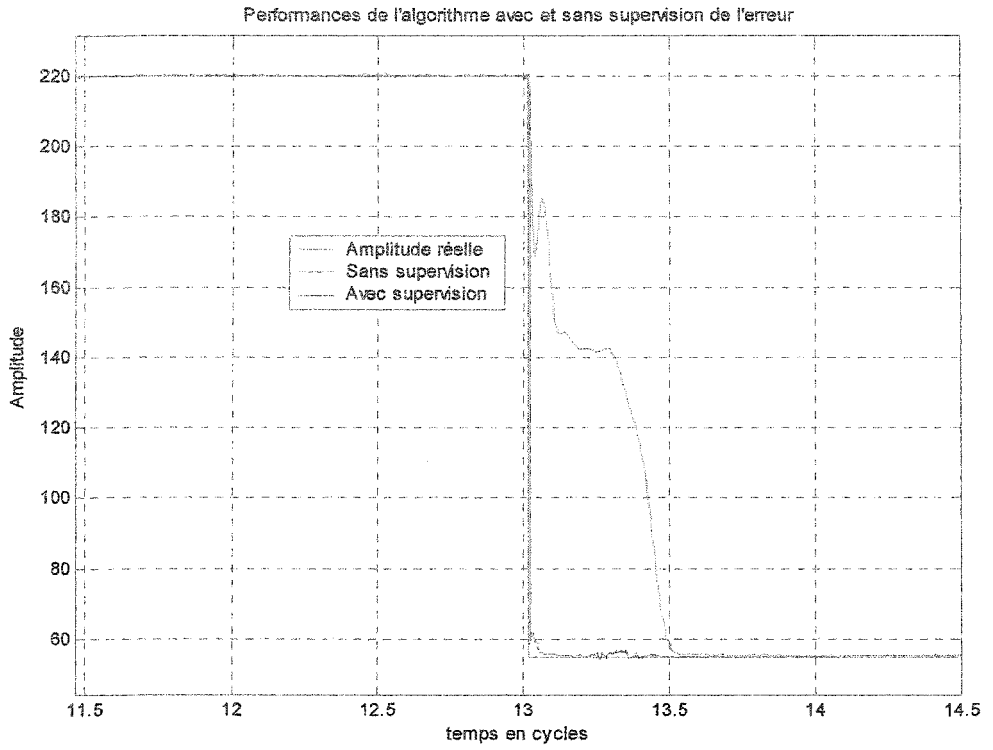


Figure 3.8 : Début du creux de tension (agrandissement de la figure 3.7)

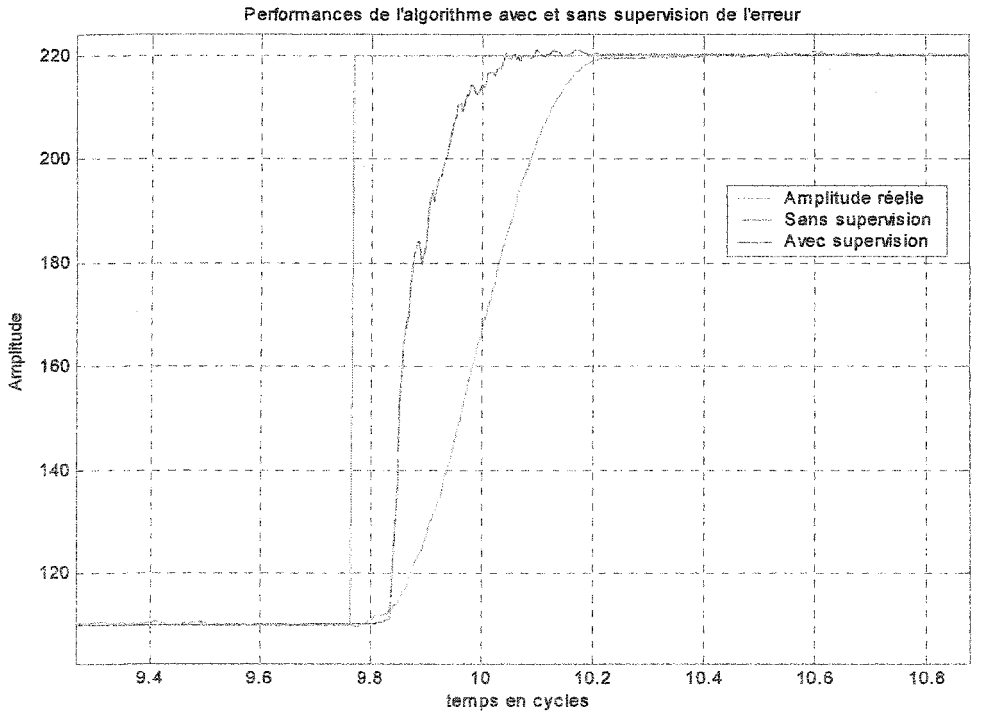


Figure 3.9 : Fin du creux de tension (agrandissement de la figure 3.7)

3.4 Conclusion

La théorie de la méthode Adaline appliquée à la détection et l'estimation des creux de tension a été exposée. Deux algorithmes ont été présentés. La précision de ces algorithmes dans l'estimation de l'amplitude du fondamental et des harmoniques d'une tension a été vérifiée en régime stationnaire. La supervision de l'erreur permet d'améliorer la précision pour l'estimation de l'amplitude du fondamental.
Les performances (rapidité) des deux algorithmes dans les régions critiques (au début et à la fin d'un creux de tension) ont été évaluées. On peut noter que la rapidité n'est pas la même au début du creux et à la fin du creux (figures 3.8 et 3.9). La supervision de l'erreur a largement contribué à améliorer la rapidité de l'algorithme Adaline de base. Au début d'un creux, la vitesse a été améliorée d'un facteur d'environ 10 et à la fin du creux, elle est améliorée d'un facteur d'environ 2 (tableau 3-3).

CHAPITRE 4 : SYSTÈME AUXILIAIRE DE COMPENSENTATION DE CREUX DE TENSION ET DES HARMONIQUES

4.1 Introduction

Le système auxiliaire de compensation de creux de tension que nous présentons permet de compenser un creux qui peut survenir aux bornes d'une charge quelconque. Il permet aussi d'améliorer la forme du courant à la source. Ce travail est une extension des travaux déjà publiés en [23] et [27]. Il s'agit d'un système composé d'un circuit principal et de deux sous-circuits auxiliaires dont l'un pour la compensation des creux de tension et l'autre pour la compensation des courants harmoniques par l'injection du troisième harmonique. Les principales contributions de ce travail sont : l'implantation en temps réel d'un algorithme rapide de détection de creux de tension basé sur la méthode Adaline, la proposition d'une loi de compenstion des creux de tension équilibrés et déséquilibrés et le développement d'un prototype de puissance de 12.5kW incluant les cartes de commande.

La topologie de ce système est présentée à la figure 4.1.

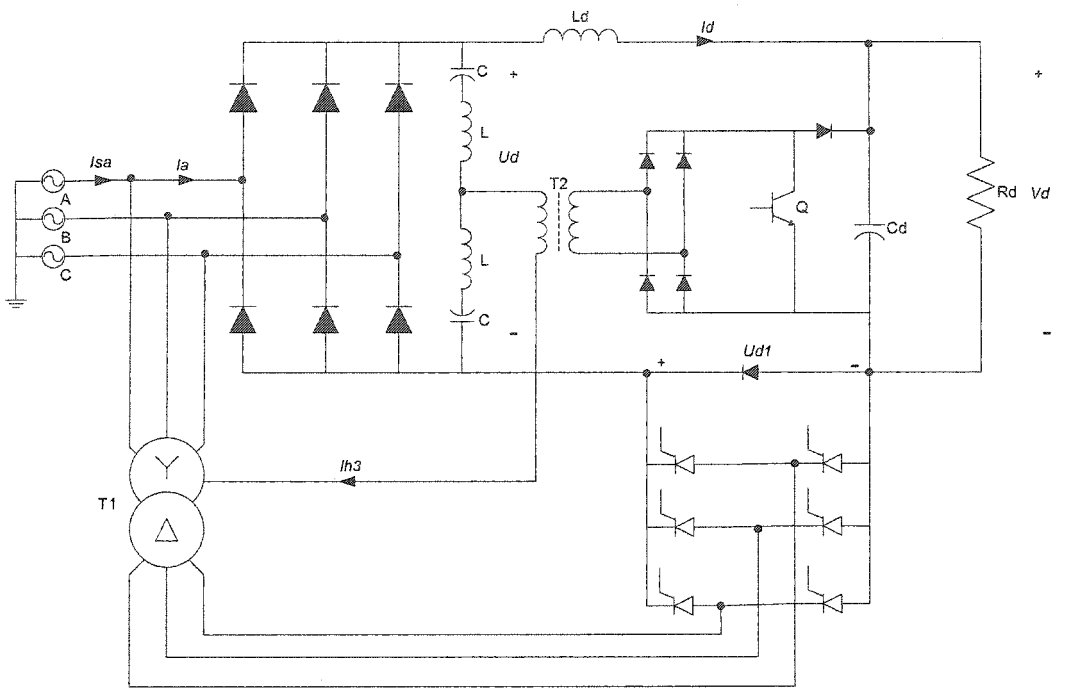


Figure 4.1 : Schéma de principe du système auxiliaire de compensation.

La solution que nous présentons dans le cadre de ce travail trouve directement son application dans les entraînements à vitesse variable (EVV). Afin d'assurer une certaine fiabilité à l'EVV durant le creux de tension, les modifications apportées à la topologie de base permettront de réguler la tension du lien cc afin que celle-ci demeure quasi-constante en présence de creux de tension.

Dans les sections qui suivent nous présenterons la théorie de la compensation des harmoniques basée sur l'injection du troisième harmonique. Des résultats de simulations seront présentés. Le principe de compensation des creux de tension équilibrés et déséquilibrés et une loi de compensation seront exposés. Des résultats de simulations de l'ensemble du système seront aussi présentés.

4.2 Compensation des harmoniques par le troisième harmonique

Les pertes et les dysfonctionnements dans les équipements sont des effets palpables causés par la pollution harmonique. Cette pollution est causée par des charges non linéaires et la majorité de ces charges sont des redresseurs [21], [22]. Elle se traduit par une déformation (distorsion) du courant de ligne.

«Compenser des harmoniques par l'injection du troisième harmonique» revient à améliorer la forme d'onde du courant de ligne; c'est-à-dire le rendre le plus sinusoïdal possible en réinjectant dans la ligne, la composante fréquentielle de 180Hz. Les courants harmoniques produits par un redresseur hexaphasé peuvent être significativement réduits en extrayant la composante du courant à 180 Hz du côté cc du redresseur et en réinjectant la quantité appropriée au côté ca. Ce principe a été appliqué pour la première fois en 1969 [23]. Elle est viable pour les applications de fortes puissances [24], [22]. Elle a l'avantage d'une faible complexité du circuit. Cette méthode a fait l'objet de nombreuses publications dans lesquelles on retrouve plusieurs variantes. Les détails et le principe de cette méthode sont bien abordés en [21], [23], [24], [25] et [26]. La théorie de la compensation des courants harmoniques dans le cadre de ce travail est présentée dans la section suivante.

4.2.1 Amélioration du taux de distorsion en courant

L'injection du troisième harmonique est une méthode qui permet de réduire la distorsion du courant d'entrée des redresseurs triphasés à diodes [24].

Considérons le redresseur à diodes de la figure 4.2 alimenté par un système triphasé balancé (v_a, v_b et v_c).

$$
\begin{aligned}
v_a &= V_m \cos(\omega_o t) \\
v_b &= V_m \cos\left(\omega_o t - \frac{2\pi}{3}\right) \\
v_c &= V_m \cos\left(\omega_o t + \frac{2\pi}{3}\right)
\end{aligned}
\tag{4.1}
$$

où V_m est l'amplitude de la tension ligne neutre et ω_b est la pulsation de la tension de ligne.

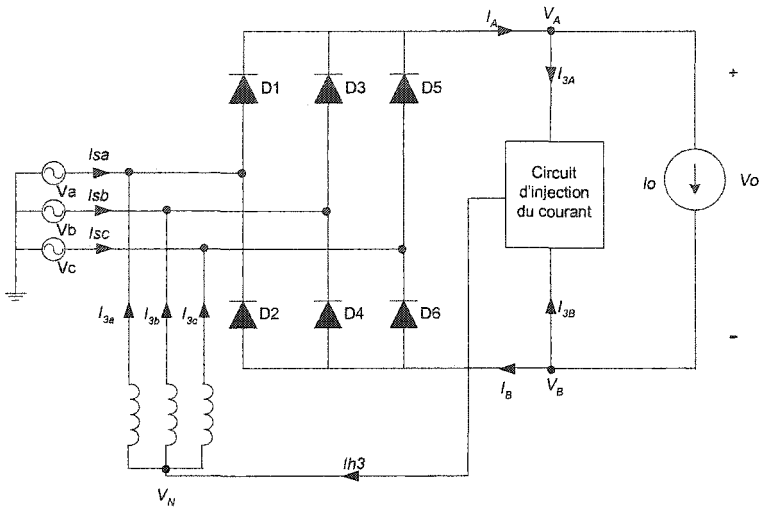


Figure 4.2 : Redresseur triphasé à diodes avec circuit d'injection.

La modélisation harmonique des redresseurs triphasés à diodes a permis de démontrer que les tensions V_A et V_B, et les courants I_A et I_B, des redresseurs côté cc sont périodiques avec une fréquence qui est le triple de la fréquence de la ligne (du réseau). Comme tout signal périodique, en appliquant la série de Fourier aux tensions et aux courants du redresseur côté cc (voir figure 4.2), on obtient [21] :

Pour les tensions V_A et V_B :

$$
V_A = \frac{3\sqrt{3}}{\pi} V_m \left(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{(1)^{n+1}}{9n^2 - 1} \cos(3n\omega_o t) \right) \tag{4.2}
$$

et

$$
V_B = \frac{3\sqrt{3}}{\pi} V_m \left(-\frac{1}{2} + \sum_{n=1}^{\infty} \frac{(1)^{n+1}}{9n^2 - 1} \cos(3n\omega_o t) \right) \tag{4.3}
$$

pour les courants I_A et I_B :

$$I_A = \frac{3\sqrt{3}}{\pi} I_m \left(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{1-2(1)^n}{9n^2-1} \cos(3n\omega_o t) \right) \tag{4.4}$$

et

$$I_B = \frac{3\sqrt{3}}{\pi} I_m \left(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{(-1)^n - 2}{9n^2-1} \cos(3n\omega_o t) \right) \tag{4.5}$$

La composante continue des courants I_A et I_B est :

$$I_o = \frac{3\sqrt{3}}{2\pi} I_m \tag{4.6}$$

Comme le démontrent les équations (4.2), (4.3), (4.4) et (4.5), les tensions et les courants côté cc comportent une composante dont la fréquence est le triple de la fréquence de la ligne. Les composantes alternatives des courants I_A et I_B transférées au circuit d'injection sont [21]:

$$I_{3A} = \frac{3\sqrt{3}}{\pi} I_m \sum_{n=1}^{\infty} \frac{1-2(1)^n}{9n^2-1} \cos(3n\omega_o t) \tag{4.7}$$

et

$$I_{3B} = \frac{3\sqrt{3}}{\pi} I_m \sum_{n=1}^{\infty} \frac{2-(-1)^n}{9n^2-1} \cos(3n\omega_o t) \tag{4.8}$$

En considérant que les impédances sur chaque phase sont égales, le courant réinjecté dans chaque phase sera [21]:

$$I_{3a} = I_{3b} = I_{3c} = \frac{1}{3} I_{h3} = \frac{1}{3}(I_{3A} + I_{3B}) \tag{4.9}$$

Un développement supplémentaire des équations (4.7) et (4.8) permet de représenter les courants I_{3A} et I_{3B} sous la forme de la somme de courants harmoniques pairs et impairs [21].

$$I_{3A} = I_{impair} + I_{pair} \tag{4.10}$$

et

$$I_{3B} = I_{impair} - I_{pair} \tag{4.11}$$

où

$$I_{impair} = \frac{3\sqrt{3}}{\pi} I_m \sum_{k=1}^{\infty} \frac{3}{9(2k-1)^2 - 1} \cos((6k-3)\omega_o t) \tag{4.12}$$

$$I_{pair} = -\frac{3\sqrt{3}}{\pi} I_m \sum_{k=1}^{\infty} \frac{1}{36k^2 - 1} \cos(6k\omega_o t) \tag{4.13}$$

Considérant (4.10) et (4.11), le courant injecté est :

$$I_{h3} = I_{3A} + I_{3B} = 2I_{impair} \tag{4.14}$$

4.2.2 Contrôle de l'injection du troisième harmonique

Pour obtenir une injection de courant optimale, le circuit d'injection de courant devrait être capable de fournir des courants tels que spécifiés par les équations (4.7) et (4.8). L'analyse des harmoniques injectés sous forme de courants harmoniques pairs et impairs a conduit à la conception d'une structure de base du circuit d'injection. Celle-ci est présentée à la figure 4.3.

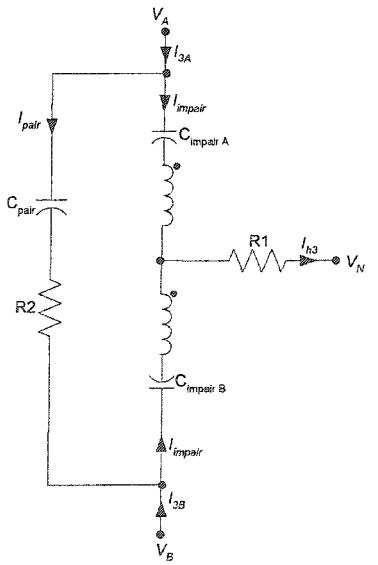


Figure 4.3 : Structure de base du circuit d'injection

Ce circuit est composé de deux réseaux qui permettent d'extraire les courants harmoniques pairs (C_{pair} et R_2) et impairs ($C_{impair\ A}$, $C_{impair\ B}$, R_1 et les deux inductances). Les formules nécessaires pour déterminer les valeurs des différents éléments du circuit d'injection sont présentées en [21]. Respectivement, les résistances R_1 et R_2 permettent de contrôler les amplitudes des courants harmoniques injectés I_{h3} et pairs [21]. Elles varient en fonction des variations de la charge de la même manière que la résistance équivalente *Re* telle que vue de la source. Les résistances R_1 et R_2 s'expriment comme suit :

$$R_1 = \frac{1}{6}\mathrm{Re} \tag{4.15}$$

$$R_2 = 2\,\mathrm{Re} \tag{4.16}$$

avec $\mathrm{Re} = \dfrac{V_m}{I_m}$

Il existe plusieurs variantes du circuit d'injection [26, 21, 23, 24, 25]; mais l'essence de ces différentes topologies demeure la même. La topologie adoptée dans le cadre de ce travail est présentée en [23] et [27] (voir figure 4.4).

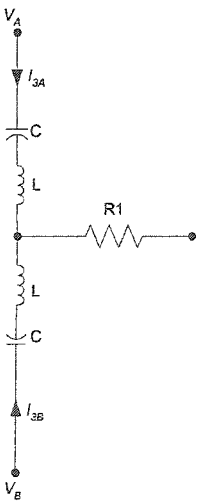


Figure 4.4 : Structure du circuit d'injection adopté

R_1 permet de contrôler les amplitudes des courants harmoniques injectés. Elle doit cependant varier en fonction des variations de la charge. Afin de rencontrer cette contrainte, R_1 a été remplacée par un élévateur de tension (boost converter) à facteur de puissance unitaire relié au reste du circuit d'injection via un transformateur [23], [27]. Ce nouvel élément permet d'émuler une résistance variable. Ainsi en fonction du rapport cyclique imposé à l'interrupteur Q de l'élévateur (voir figure 4.5), l'amplitude du troisième harmonique réinjecté dans la ligne sera contrôlée.

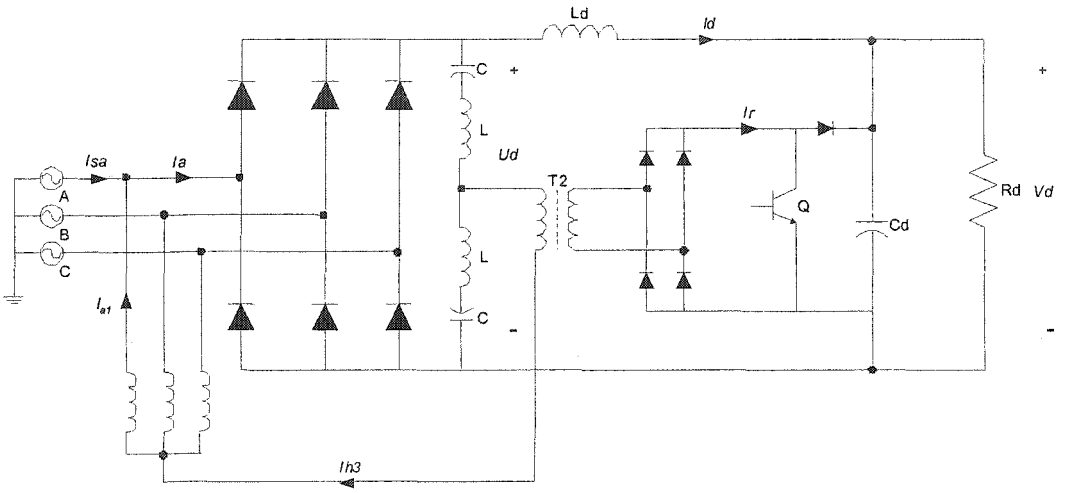


Figure 4.5 : Redresseur triphasé avec circuit d'injection du 3e harmonique

Le circuit de commande du contrôle du troisième harmonique est proposé en [23]. Son schéma est présenté à la figure 4.6.

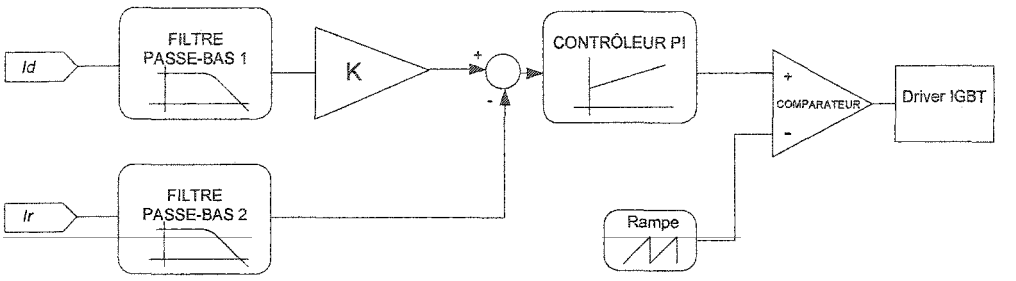


Figure 4.6 : Commande du contrôle du circuit d'injection

Les filtres passe-bas 1 et 2 sont respectivement d'ordre 2 et 4. Ils ont un gain unitaire et une fréquence de coupure de 70Hz. Le gain K est égal à 0.16.

Le régulateur PI a un gain proportionnel $K_P = 2$ et un gain intégral $K_I = 20$. Le générateur de rampe a une fréquence de 5kHz.

La section suivante présente les simulations effectuées avec le logiciel PSIM. L'amélioration du taux de distorsion harmonique apportée par la méthode d'injection du troisième harmonique sera présentée.

4.2.3 Résultats de simulation

Nous avons considéré le schéma de la figure 4.1. Lors des simulations, le redresseur commandé à thyristors est hors fonction. Les résultats de simulation ont été obtenus à l'aide du logiciel PSIM.

Les paramètres de simulation sont les suivants :

Source d'alimentation : $208V_{LL}$, 60 Hz

Transformateur T_1 : 10kVA, 208V/208V

Transformateur T_2 : 1.5kVA, 35V/210V

Filtre LC : L = 11.7mH, C = 60μF, $r = 0.2\Omega$

Inductance de lissage : L_d = 10mH;

Condensateur : C_d = 1100μF;

Charge : R_d = 6.2Ω (soit P = 12.5kW).

La figure 4.7 présente le courant dans la ligne I_{sa} (à la source) sans le système auxiliaire d'injection du troisième harmonique. L'axe des ordonnés est en ampère.

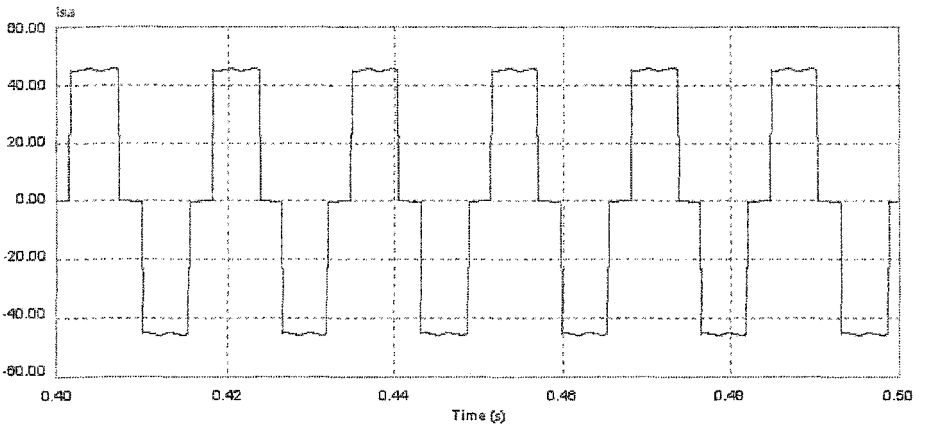


Figure 4.7 : Courant dans la ligne sans le circuit d'injection du 3e harmonique

La figure 4.8 présente le courant dans la ligne I_{sa} avec le système auxiliaire d'injection du troisième harmonique. L'axe des ordonnés est en ampère.

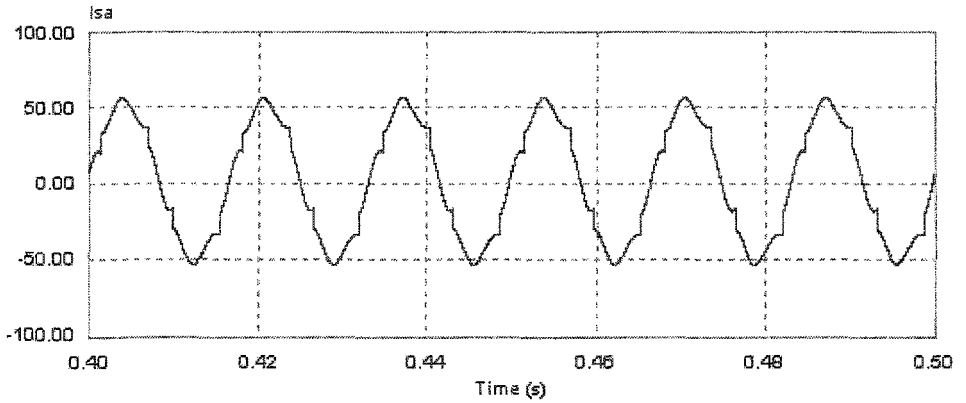


Figure 4.8 : Courant dans la ligne avec le circuit d'injection du 3e harmonique

L'application de la FFT sur les courants de ligne des différentes phases nous a permis de calculer les taux individuels de distorsion en courant pour les harmoniques de rang 5, 7, 11, 13, 17, 19 et 23. Ces valeurs exprimées en pourcentage du fondamental du courant (valeur efficace), sont consignées dans les tableaux 4-1 et 4-2.

Sans le circuit d'injection, les valeurs efficaces des courants dans les phases A, B et C sont respectivement de 50.00A, 50.03A et 49.90A. Avec le circuit d'injection, ces courants sont de 52.13A, 52.14A et 52.14A.

Tableau 4-1 : Taux individuel d'harmonique de courant sans le circuit d'injection du 3e harmonique

	Hn (%)		
Rang harmonique	***Phase A***	***Phase B***	***Phase C***
5	20,14	19,93	20,28
7	14,06	14,25	14,03
11	9,06	8,87	9,16
13	7,50	7,72	7,47
17	5,82	5,62	5,93
19	5,08	5,30	5,05
23	4,26	4,08	4,37

Tableau 4-2 : Taux individuel d'harmonique de courant avec le circuit d'injection du 3e harmonique

	Hn (%)		
Rang harmonique	***Phase A***	***Phase B***	***Phase C***
5	3,59	3,61	3,61
7	1,36	1,35	1,36
11	1,50	1,50	1,50
13	1,42	1,42	1,42
17	1,17	1,15	1,17
19	1,06	1,05	1,07
23	0,90	0,88	0,90

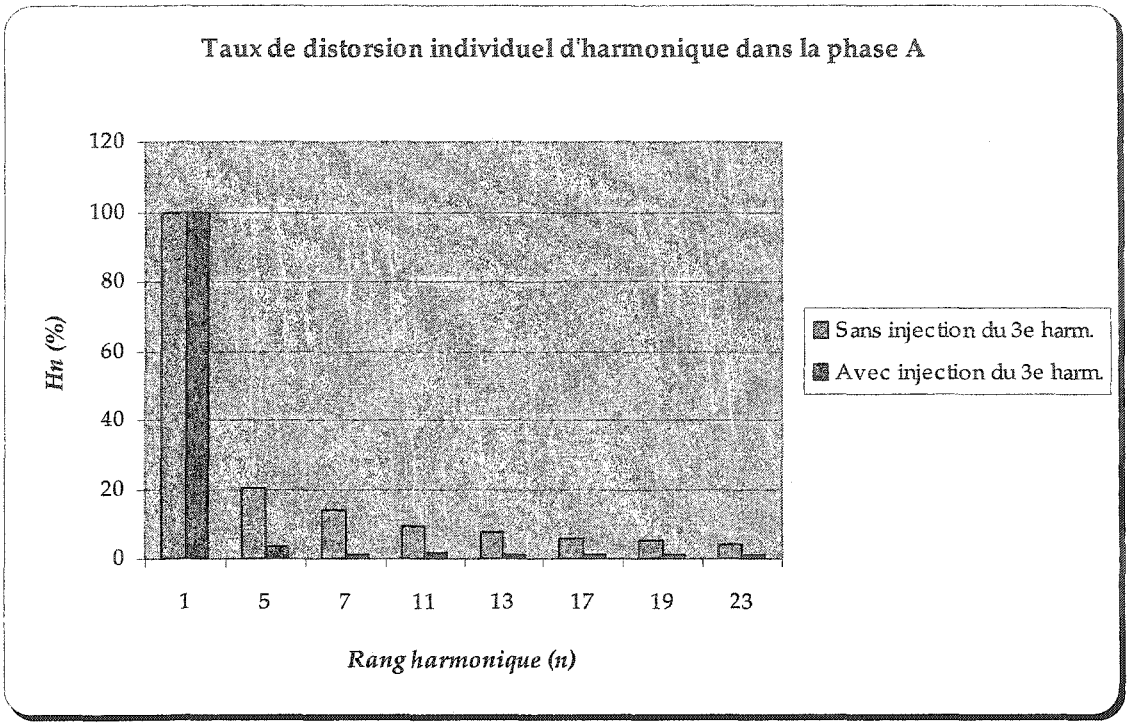


Figure 4.9 : Taux individuel d'harmonique de courant la phase A

Le tableau 4-3 présente le taux global de distorsion dans chacune des phases.

Tableau 4-3 : Taux de distorsion harmonique global dans les différentes phases

	THD dans les trois phases (%)		
	Phase A	*Phase B*	*Phase C*
Sans compensation	28,63	28,54	28,77
Avec compensation	4,72	4,72	4,74

Il a été démontré que les harmoniques côté ca sont induits par les harmoniques côté cc [13]. Afin d'évaluer les performances de la méthode d'injection du troisième harmonique, nous avons trouvé intéressant de mesurer le taux de distorsion harmonique individuel du courant de ligne en fonction de la valeur de l'inductance de lissage L_d. Pour cela, nous avons fait varier l'inductance de lissage de 1mH à 40mH. Les valeurs obtenues sont représentées sur le graphique de la figure 4.10.

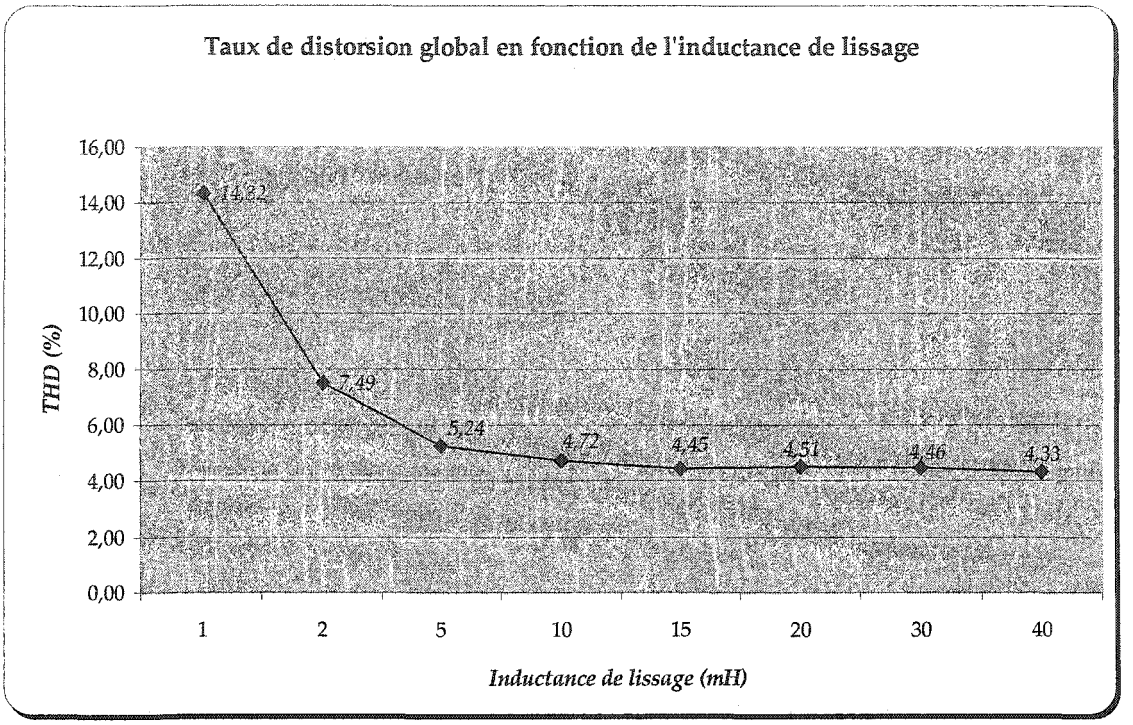


Figure 4.10 : Taux de distorsion harmonique dans la ligne en fonction de Ld

Nous avons effectué un ensemble de simulations nous permettant de valider les performances du circuit d'émulation d'une résistance (élévateur de tension). Rappelons-le, ce circuit permet d'optimiser l'injection du troisième harmonique en contrôlant l'amplitude du troisième harmonique réinjecté. Or l'amplitude du troisième harmonique peut varier en fonction de la valeur de la résistance de charge (R_d) soit la puissance de sortie. Raison pour laquelle nous avons mesuré le THD dans la phase A pour différentes valeurs de la résistance de charge équivalentes à différentes valeurs de puissance. Les résultats obtenus sont représentés dans le graphique de la figure 4.11.

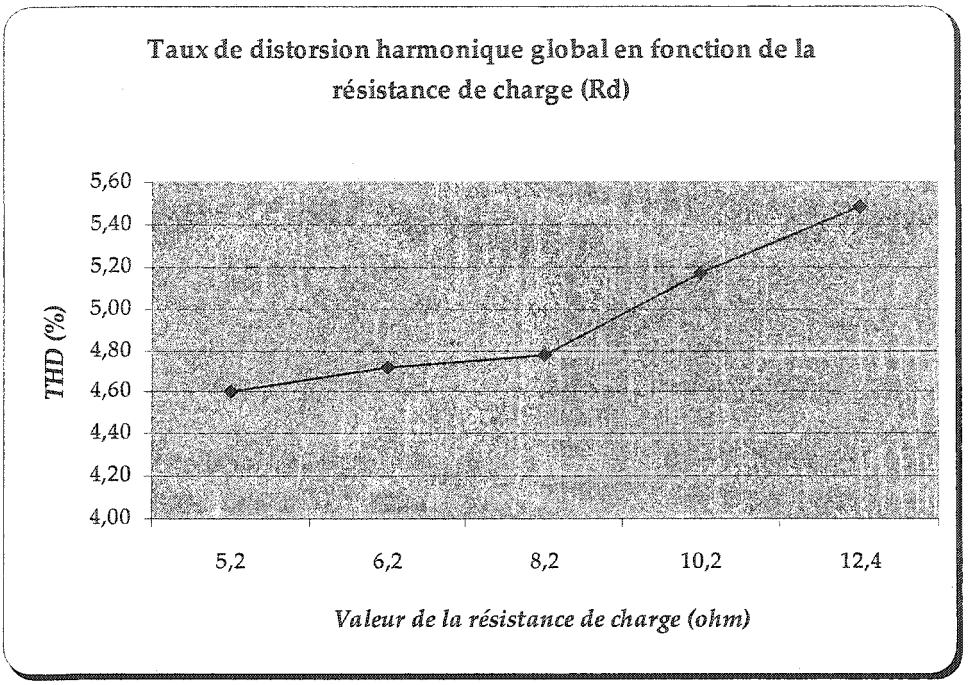


Figure 4.11 : Taux de distorsion harmonique dans la ligne en fonction de R_d

Ces deux ensembles de résultats démontrent que la méthode d'injection du troisième harmonique permet certes d'améliorer grandement le THD mais, ces performances sont tributaires de l'inductance de lissage et de la résistance de charge (puissance de sortie).

4.3 Compensation du creux de tension

4.3.1 Principe de compensation

Un redresseur commandé à thyristors avec diode de roue libre en sortie est inséré en série entre le redresseur à diodes et la charge tel que présenté à la figure 4.12.

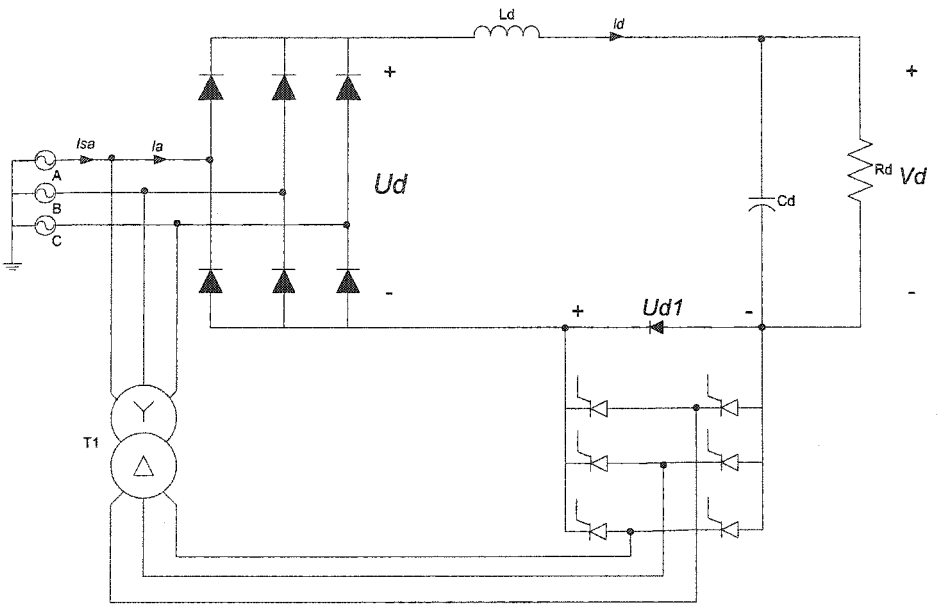


Figure 4.12 : Principe de compensation des creux de tension.

Dans les conditions de creux de tension, le redresseur commandé permettra de maintenir la tension du lien cc, soit V_d quasi-constante. Pour exposer plus en détails le principe de compensation, le circuit de compensation du troisième harmonique est négligé (voir figure 4.12) et nous supposons aussi que l'inductance L_d est assez grand pour garder le redresseur dans un mode de conduction continue.

En première approximation, la tension aux bornes de la charge peut être écrite de la manière suivante [27]:

$$V_d = U_d + U_{d1} \tag{4.17}$$

Pendant l'absence d'un creux de tension, le redresseur à thyristors est hors service et la diode de roue libre (à la sortie du redresseur) conduit. La tension U_{d1} est nulle. La valeur moyenne de la tension du redresseur principal (redresseur à diodes) est :

$$U_{do} = \frac{3\sqrt{3}}{2\pi} V_{LN} \left(1 + \cos\mu\right) \tag{4.18}$$

V_{LN} est l'amplitude de la tension ligne-neutre;

μ est l'angle d'empiètement.

Lorsque le creux de tension est détecté, le pont à thyristors entre en fonctionnement et bloque la diode de roue libre.

La tension moyenne, U_{d1o}, fournie par le redresseur à thyristors auxiliaire est exprimée comme suit :

$$\begin{cases} U_{d1o} = \dfrac{U_{do}}{\sqrt{3}K_{T1}} \cos\alpha & si \quad 0 \le \alpha < \dfrac{\pi}{3} \\ U_{d1o} = \dfrac{U_{do}}{\sqrt{3}K_{T1}} \left[1 + \sin\left(\dfrac{\pi}{6} - \alpha\right)\right] & si \quad \dfrac{\pi}{3} < \alpha < \dfrac{2\pi}{3} \\ U_{d1o} = 0 & si \quad \alpha > \dfrac{2\pi}{3} \end{cases} \tag{4.19}$$

K_{T1} est le rapport de transformation du transformateur T_1.

4.3.2 Loi de commande

Notre loi de commande repose sur la compensation de la tension moyenne aux bornes de la charge. Ayant déterminé la diminution de la tension moyenne ΔU_{do} dans le lien cc, la question qui se pose est de déterminer l'angle d'amorçage α à imposer au redresseur auxiliaire afin d'obtenir la même valeur moyenne que la

chute ΔU_{do} du redresseur principal. En présence d'un creux de tension, la tension moyenne U_{d1o} du redresseur auxiliaire à fournir devrait donc être égale à ΔU_{do}.

Définissons une variable K_o comme suit :

$$K_o = \frac{(U_{do} - U_{d1o})}{U_{do}} \tag{4.20}$$

En présence de creux de tension, (4.20) nous donne :

$$K_o = \frac{(U_{do} - \Delta U_{do})}{U_{do}} \tag{4.21}$$

ou encore

$$K_o = 1 - \frac{\Delta U_{do}}{U_{do}} \tag{4.22}$$

$$U_{d1o} = \Delta U_{do} = (1 - K_o) U_{do} \tag{4.23}$$

(4.19) devient
$$\begin{cases} 1 - K_o = \dfrac{1}{\sqrt{3} K_{T1}} \cos\alpha & si \quad 0 \le \alpha < \dfrac{\pi}{3} \\ 1 - K_o = \dfrac{1}{\sqrt{3} K_{T1}} \left[1 + \sin\left(\dfrac{\pi}{6} - \alpha \right) \right] & si \quad \dfrac{\pi}{3} < \alpha < \dfrac{2\pi}{3} \end{cases}$$

$$\begin{cases} K_o = 1 - \dfrac{1}{\sqrt{3} K_{T1}} \cos\alpha & si \quad 0 \le \alpha < \dfrac{\pi}{3} \\ K_o = 1 - \dfrac{1}{\sqrt{3} K_{T1}} \left[1 + \sin\left(\dfrac{\pi}{6} - \alpha \right) \right] & si \quad \dfrac{\pi}{3} < \alpha < \dfrac{2\pi}{3} \end{cases} \tag{4.24}$$

L'équation (4.24) nous donne une expression : $K_o = f(\alpha)$ avec $0 \le K_o \le 1$. Il devient facile de déduire l'angle α en appliquant l'inverse de la fonction appropriée. En pratique, l'emploi des tables deux dimensions est très courant pour éviter la difficulté inhérente à l'implantation de l'inverse des fonctions trigonométriques. Connaissant la plage de valeurs de K_o, il devient aisé de calculer les angles correspondants aux différentes valeurs de K_o.

4.4 Résultats de simulation

Le schéma PSIM du système auxiliaire de compensation est présenté à la figure 4.13.

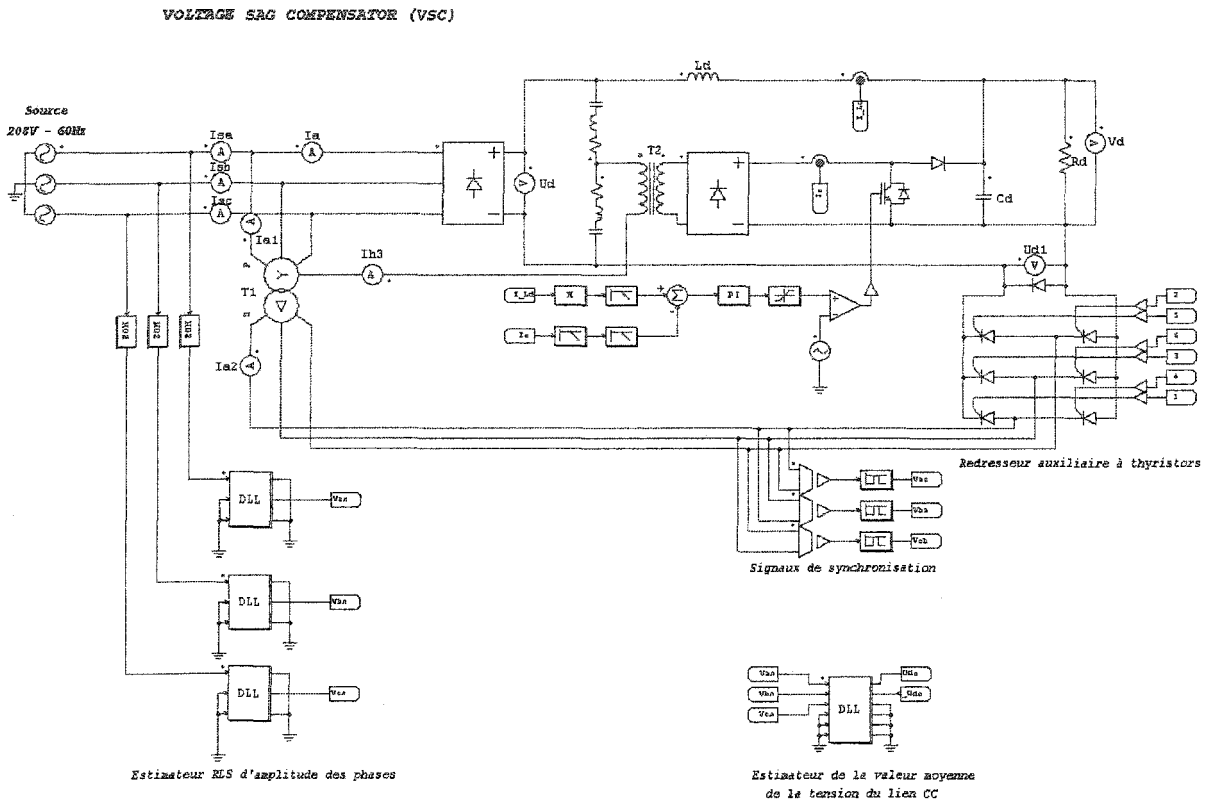


Figure 4.13 : Schéma PSIM du système auxiliaire de compensation de creux de tension.

Le point de mesure pour une détection rapide des creux de tension est à la source côté ca.

Trois estimateurs fournissent à chaque instant les valeurs des amplitudes des trois phases. Il s'agit de trois algorithmes identiques codés en langage C. Ces algorithmes sont présentés au chapitre 3. Le code en langage C est présenté à l'annexe B.

Grâce aux équations développées au chapitre 2, les trois amplitudes estimées permettront de calculer la chute de la tension moyenne dans le lien cc lors d'un creux de tension.

La figure 4.14 présente le schéma PSIM du circuit de commande du redresseur à thyristors. Il s'agit d'une commande individuelle. Dans cette commande, les impulsions d'allumage des thyristors sont référencées par rapport au passage par zéro des tensions de ligne v_{ac}, v_{ba}, v_{cb}, v_{ca}, v_{ab} et v_{bc}. Cette commande a été retenue par rapport à la commande équidistante parce qu'en régime déséquilibré, il est plus facile de prédire la tension moyenne pour un angle donné. Cette assertion peut se justifier par le fait que les performances des redresseurs sont tributaires de la méthode d'allumage employée [12].

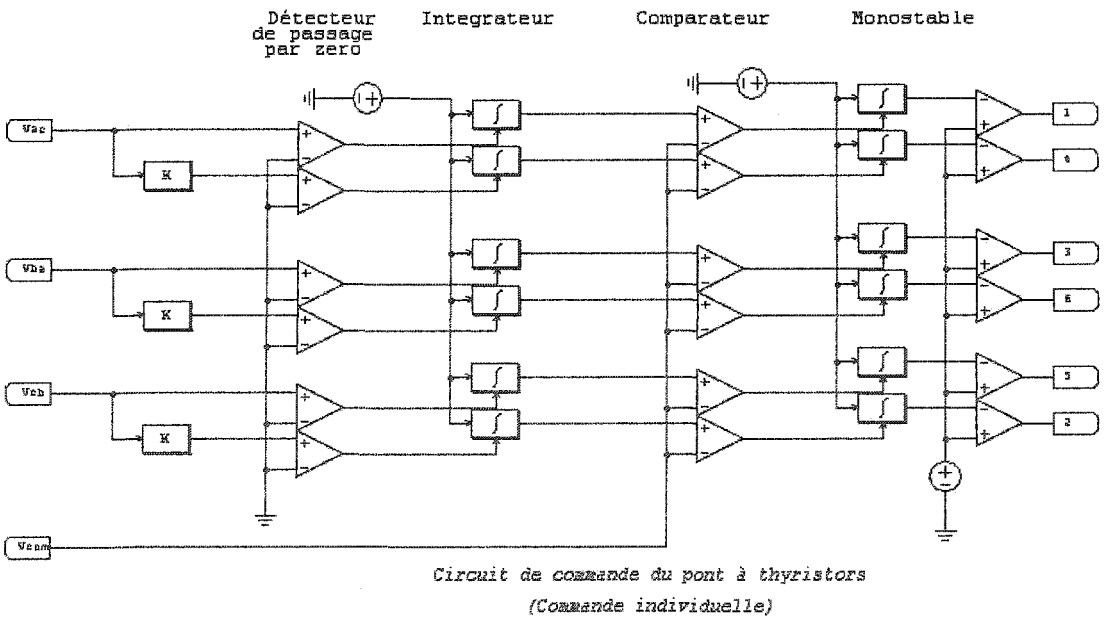


Figure 4.14 : Commande du redresseur à thyristors.

La figure 4.15 présente le schéma PSIM de la loi de commande exposée à la section 4.3.2. Le bloc *Alpha = f(Ko)* est une table à deux dimensions qui contient les valeurs

des angles d'amorçage précalculés tels que présentés à la section 4.3.2 en fonction de Ko ($0 \leq K_o \leq 1$).

Le bloc K est un gain qui permet de convertir l'angle d'amorçage en degré en tension de commande.

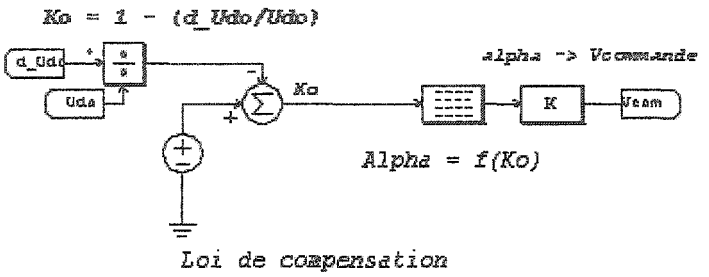


Figure 4.15 : Implantation de la loi de compensation sous PSIM.

Les résultats de simulation présentés dans cette section illustrent les performances de l'ensemble du système en présence de creux équilibrés et déséquilibrés.

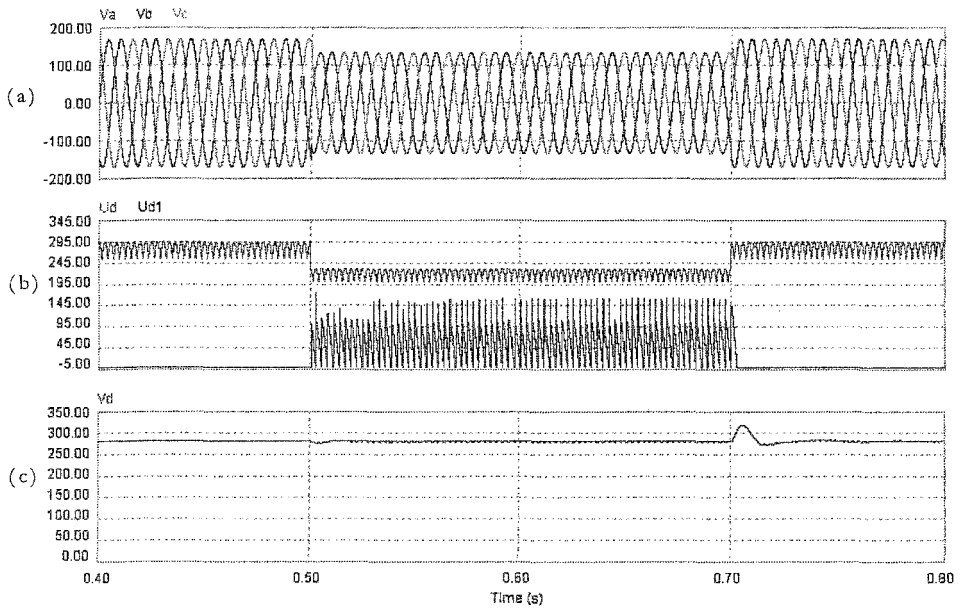


Figure 4.16 : Creux de tension de 30% de type A pendant 0.2 seconde.

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

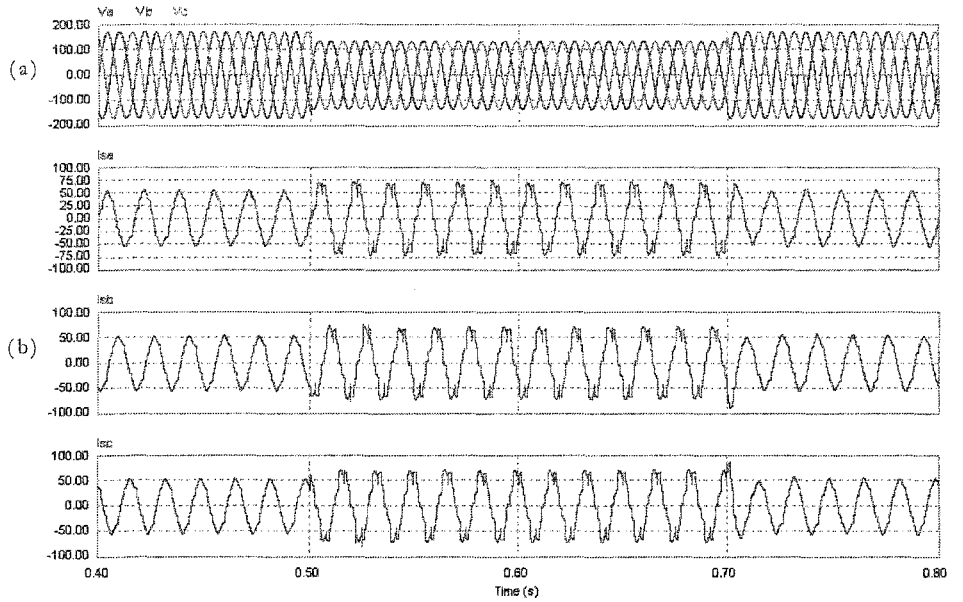


Figure 4.17 : Courants de ligne lors d'un creux de tension de 30% de type A

a) Tensions triphasées de la source
b) Courants des phases A, B et C

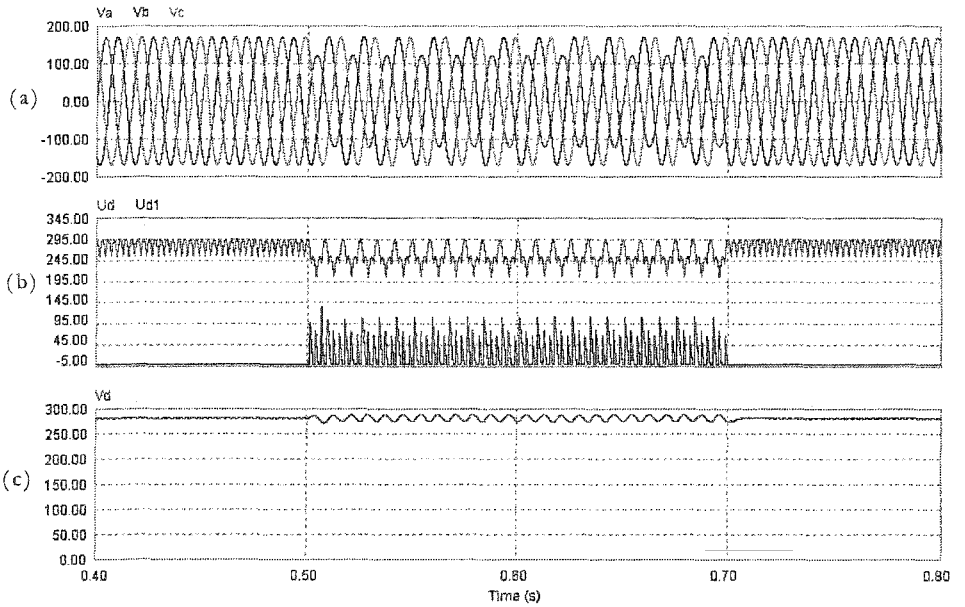


Figure 4.18 : Creux de tension de 40% de type B pendant 0.2 seconde

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

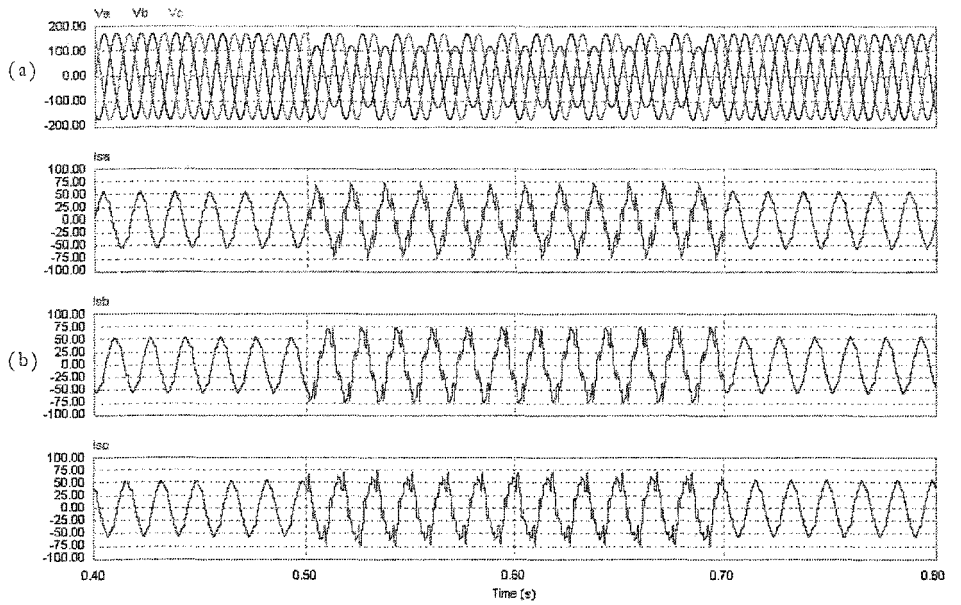


Figure 4.19 : Courants de ligne lors d'un creux de tension de 40% de type B

a) Tensions triphasées de la source
b) Courants des phases A, B et C

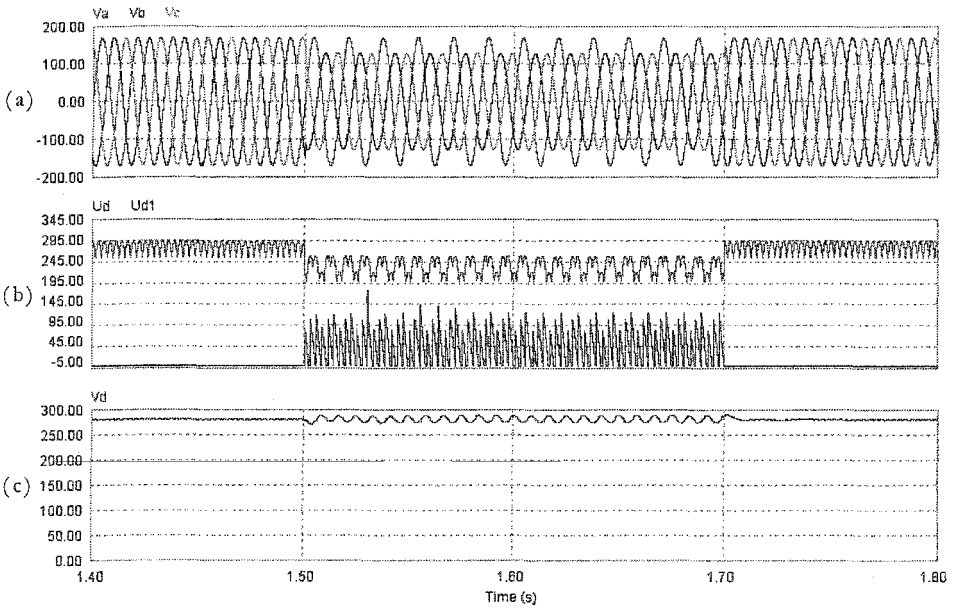


Figure 4.20 : Creux de tension de 35% de type C pendant 0.2 seconde

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

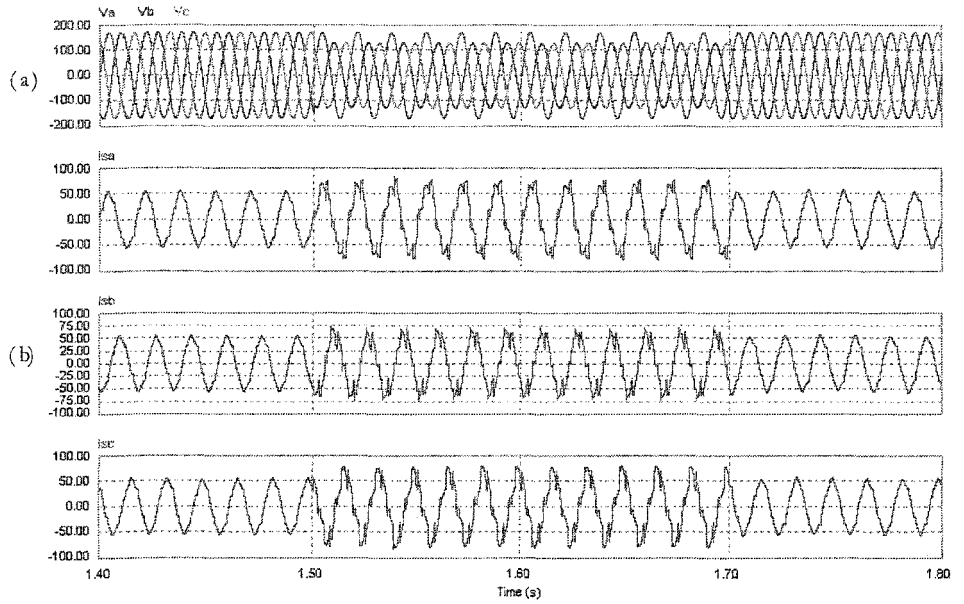


Figure 4.21 : Courants de ligne lors d'un creux de tension de 35% de type C

a) Tensions triphasées de la source
b) Courants des phases A, B et C

Afin de vérifier la robustesse de notre loi de compensation par rapport aux variations de la résistance de la charge R_d, nous avons fait varier la résistance R_d de 50% à 200% de la valeur nominale qui est de 6.2Ω.

Nous avons considéré un creux de tension de type B, soit une faute sur la phase *a* de la source entraînant une diminution de la tension de 47.13%. Les courbes suivantes sont celles obtenues pour les valeurs de charge de 12.4Ω, 6.2Ω et 3.1Ω.

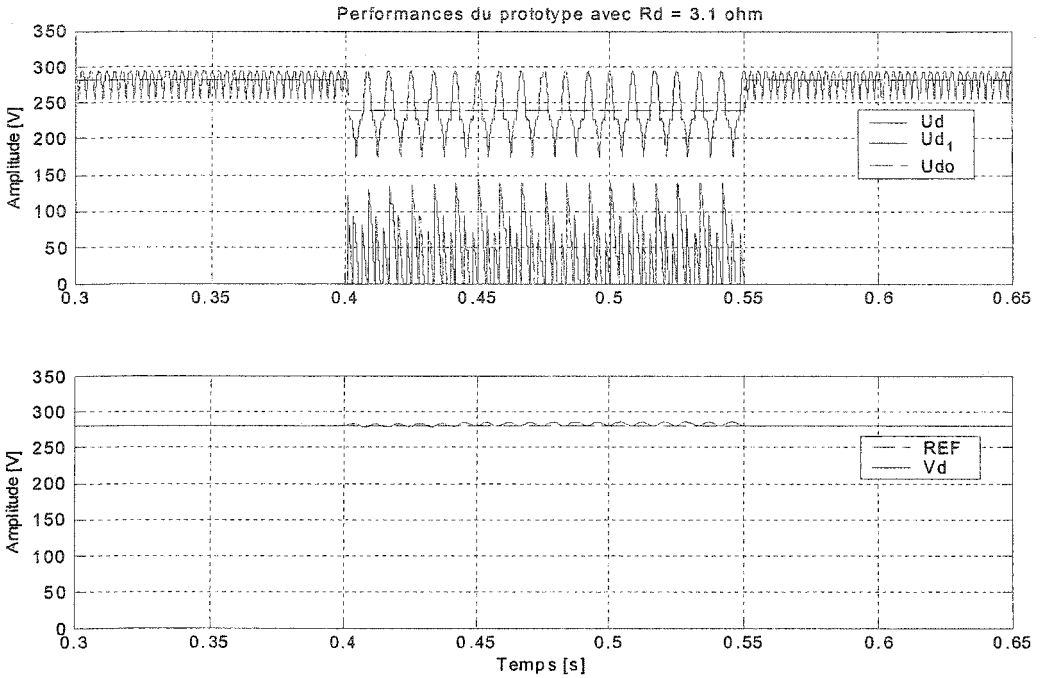


Figure 4.22 : Performances du système auxiliaire avec Rd=3.1 ohms

a) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire
b) V_d : Tension aux bornes de la charge

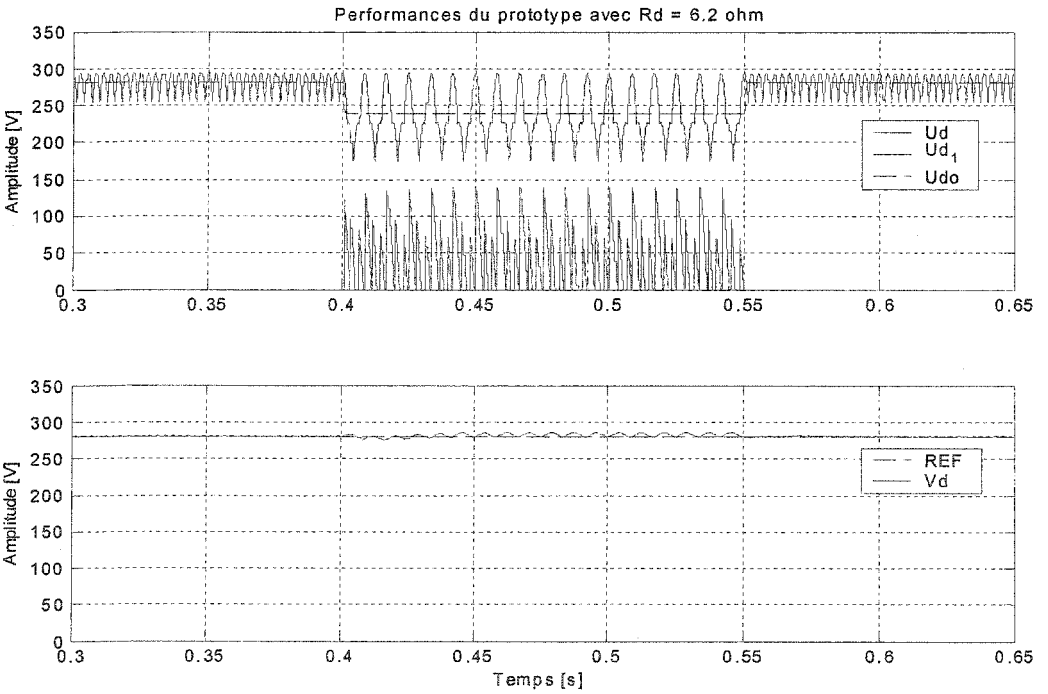


Figure 4.23 : Performances du système auxiliaire avec Rd=6.2 ohms

a) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire
b) V_d : Tension aux bornes de la charge

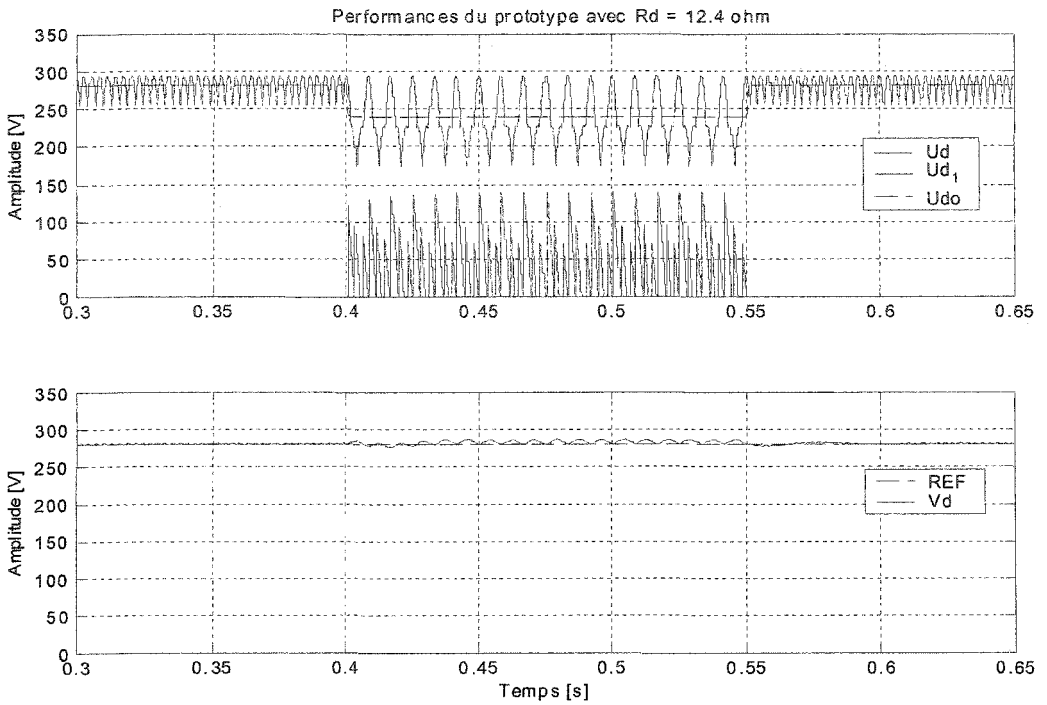


Figure 4.24 : Performances du système auxiliaire avec Rd=12.4 ohms

a) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire
b) V_d : Tension aux bornes de la charge

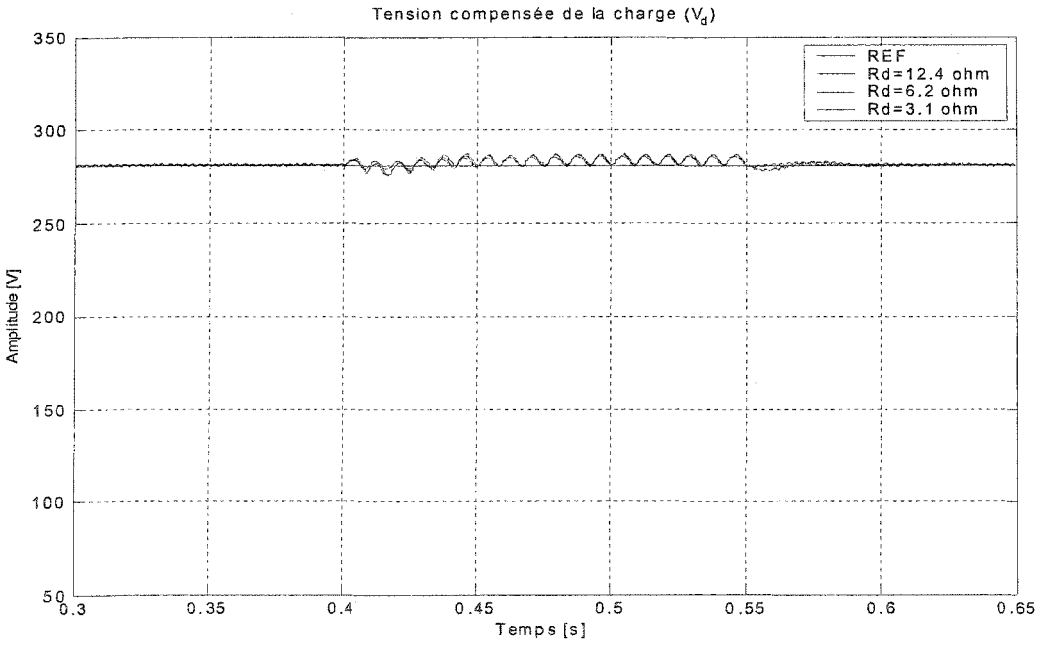


Figure 4.25 : Tension aux bornes de la charge pour trois valeurs de charge.

4.5 Conclusion

Dans ce chapitre, nous avons présenté les modifications apportées à un équipement électrique avec redresseur en pont de diodes afin de le désensibiliser aux effets de creux de tension, ce en régulant la tension de son lien cc en présence de creux de tension équilibrés et déséquilibrés. Dans le but d'améliorer aussi la forme du courant de la source (le rendre le plus sinusoïdal possible), la méthode d'injection du troisième harmonique a été exposée et appliquée. Cette méthode ne permet pas hélas d'éliminer complètement tous les harmoniques. Malgré l'extrême simplicité de son circuit, elle permet tout de même d'obtenir une réduction substantielle du taux global de distorsion de l'ordre de 83%. Un bon choix de l'inductance de lissage est nécessaire afin d'obtenir des performances optimales (voir figure 4.10). Les perfomances de la méthode d'injection du troisième harmonique sont tributaires de la valeur de l'inductance de lissage. Plus la valeur de cette inductance est élevée, meilleur est le THD du courant de ligne.

L'analyse des convertisseurs faite au chapitre 2 nous a permis de bâtir une loi de commande basée sur la compensation de la valeur moyenne de la tension pour des creux de tensions équilibrés et déséquilibrés. La comparaison des trois courbes de la tension de charge (figure 4.22 à 4.24) nous permet de constater que la loi de compensation basée sur la compensation de la valeur moyenne est fiable même pour une grande plage de variation de la résistance de charge.

CHAPITRE 5 : RÉALISATION ET VALIDATION EXPÉRIMENTALE DU PROTOTYPE

5.1 Introduction

Ce chapitre présente la réalisation du système auxiliaire de compensation simulé au chapitre 4. Dans les sections suivantes, nous commencerons par une présentation sommaire de l'environnement expérimental. L'implantation des algorithmes de détection de creux de tension et de la loi de compensation et la conception des divers circuits électroniques de commande seront exposées. Les résultats expérimentaux sont également présentés.

5.2 Présentation de l'environnement expérimental

Notre environnement expérimental peut être divisé en quatre grandes parties :

- **Les convertisseurs de puissance** : Il s'agit de l'ensemble constitué du redresseur à diodes, du redresseur commandé à thyristors et du circuit d'injection du troisième harmonique.
- **Les cartes de commande** : Il s'agit de la carte utilisée pour la commande du redresseur auxiliaire et celle utilisée pour la commande de l'interrupteur Q (voir figure 5.1).
- **La source programmable** : Elle a une puissance de 30kVA et peut débiter un courant de 250A. En mode programmé, cette source est en mesure de fournir des tensions de forme complexes; c'est-à-dire aussi bien des tensions non sinusoïdales que des tensions avec des amplitudes variables.

- **Le système dSPACE** : Deux systèmes de ce type sont utilisés dans notre expérimentation. Le premier système permet de générer les signaux de commande (signaux de faible puissance) pour la source programmable. L'ensemble formé de dSPACE 1 et de la source programmable constitue un générateur de creux de tension (*sag generator*) nécessaire pour tester la sensibilité des équipements électriques (par exemple les EVV) face aux creux de tension. Le deuxième système (dSPACE 2) permet d'implanter l'estimateur d'amplitude, l'estimateur de la valeur moyenne ainsi que la loi de compensation des creux de tension (figure 5.1).

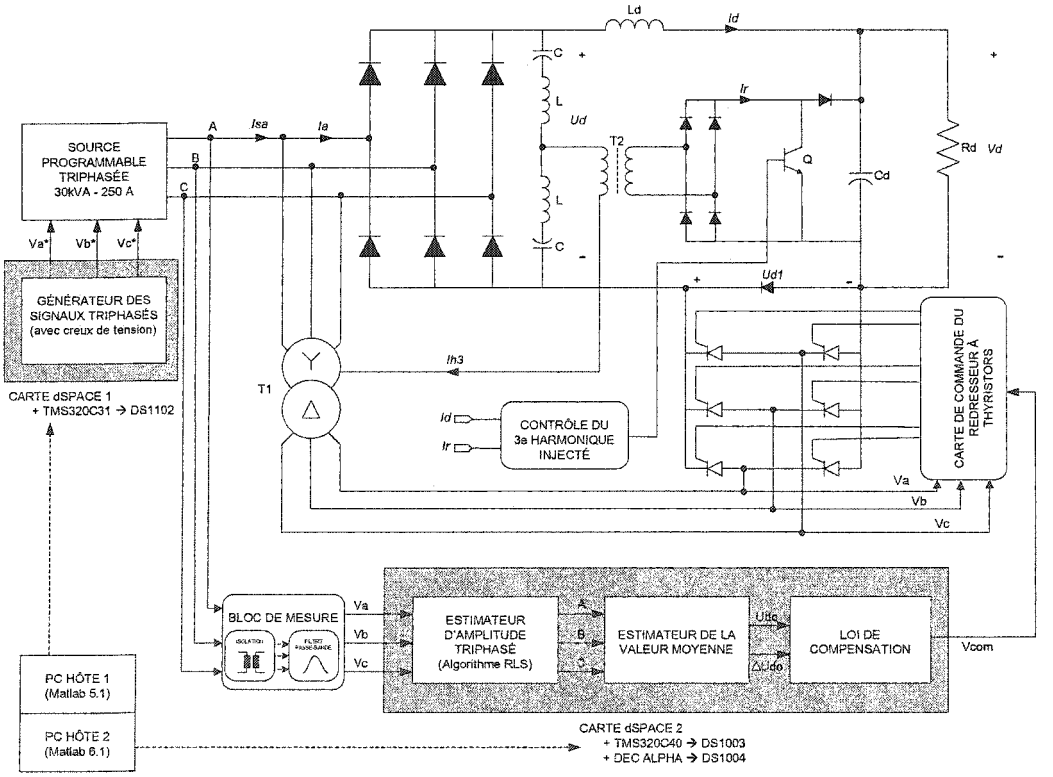


Figure 5.1 : Schéma d'intégration du prototype avec dSPACE

Les deux premières parties ayant été présentées dans les chapitres précédents, donnons un aperçu du système dSPACE.

5.2.1 Présentation de l'environnement dSPACE

Le système dSPACE constitue une plate-forme matérielle et logicielle très répandue dans de nombreuses industries dont l'industrie de l'automobile et l'aérospatiale. Il s'agit d'un système de contrôle numérique basé sur une carte de processeur numérique du signal (DSP) de Texas Instrument. Il est directement interfacé avec MATLAB/Simulink s'exécutant sur un PC. Une caractéristique intéressante du système est qu'il permet de développer un schéma de simulation ou de commande dans l'environnement Simulink, de convertir les modèles Simulink en code C et de télécharger ce code compilé sur des cartes DSP qui sera ensuite exécuté en temps réel en interagissant avec un ou d'autres systèmes physiques externes via une carte d'E/S (ADC, DAC, PWM, CAPTURE, etc.).

Deux composantes logicielles permettent la génération automatique de code : Real-Time Workshop (RTW) et Real-Time Interface (RTI). RTW permet de générer du code C à partir des modèles Simulink. Il construit automatiquement des programmes qui s'exécutent en temps réel ou sous forme de simulations « hors-ligne » (off-line). Le code généré peut s'exécuter sur du matériel PC, des DSP ou des microcontrôleurs. RTI réalise la même fonction que RTW. Il permet de compiler les blocs d'E/S. RTI est une extension de RTW mais son usage est spécifique aux cartes installées. Par exemple, pour la carte DS1102, on aura besoin de RTI1102 alors que pour une carte DS1003, RTI1003 est requise.

Bien que la plate-forme logicielle des systèmes dSPACE demeure la même (à l'exception des différences relatives à la version de MATLAB ou de RTI), la configuration matérielle elle, peut varier d'un système à un autre. Prenons l'illustration dans le cadre de notre projet; notre premier système (dSPACE 1) est doté d'une carte DS1102. Il s'agit d'une carte DSP de 32 bits de Texas Instrument, le TMS320C31 cadencé à 40 MHz. Le deuxième système possède deux cartes (système

multiprocesseur) : La carte DS1003 bâtie autour du TMS320C40 cadencé à 50 MHz et la carte DS1004 bâtie autour d'un processeur DEC Alpha 21164 cadencé à 500MHz.

Il est intéressant de présenter trois outils de contrôle de l'environnement dSPACE, à savoir :

- TRACE qui permet de visualiser à l'écran, l'évolution de certaines grandeurs caractéristiques du système commandé, un peu à l'image d'un oscilloscope.
- COCKPIT qui constitue le tableau de bord de l'application permettant ainsi de visualiser et d'agir sur les paramètres de réglage du système commandé.
- ControlDesk est l'outil de contrôle présent sur les systèmes récents (dSPACE 2). Il intègre les deux premiers outils précédemment cités qu'on rencontre sur les systèmes moins récents comme dSPACE 1.

5.2.2 Commande de la source programmable

La carte DS1102 (dSPACE 1) est utilisée pour implanter un modèle Simulink qui permet de générer des signaux triphasés. Ces signaux sont appliqués à la source programmable via le convertisseur numérique analogique de la carte DS1102.
Grâce à l'outil COCKPIT une interface de contrôle a été développée et permet d'appliquer un creux à un instant choisi. Le type de creux de tension (A, B, ou C) peut être appliqué conformément aux paramètres fournis. Il est donc possible de préciser l'amplitude et la durée du creux sur chacune des phases.

5.2.3 Implantation des estimateurs et de la loi de commande sous dSPACE

Cette implantation se fait dans le système dSPACE 2. Ce système possède deux cartes : les cartes DS1003 et DS1004. La figure 5.2 présente la structure de l'implantation dans le système dSPACE 2.

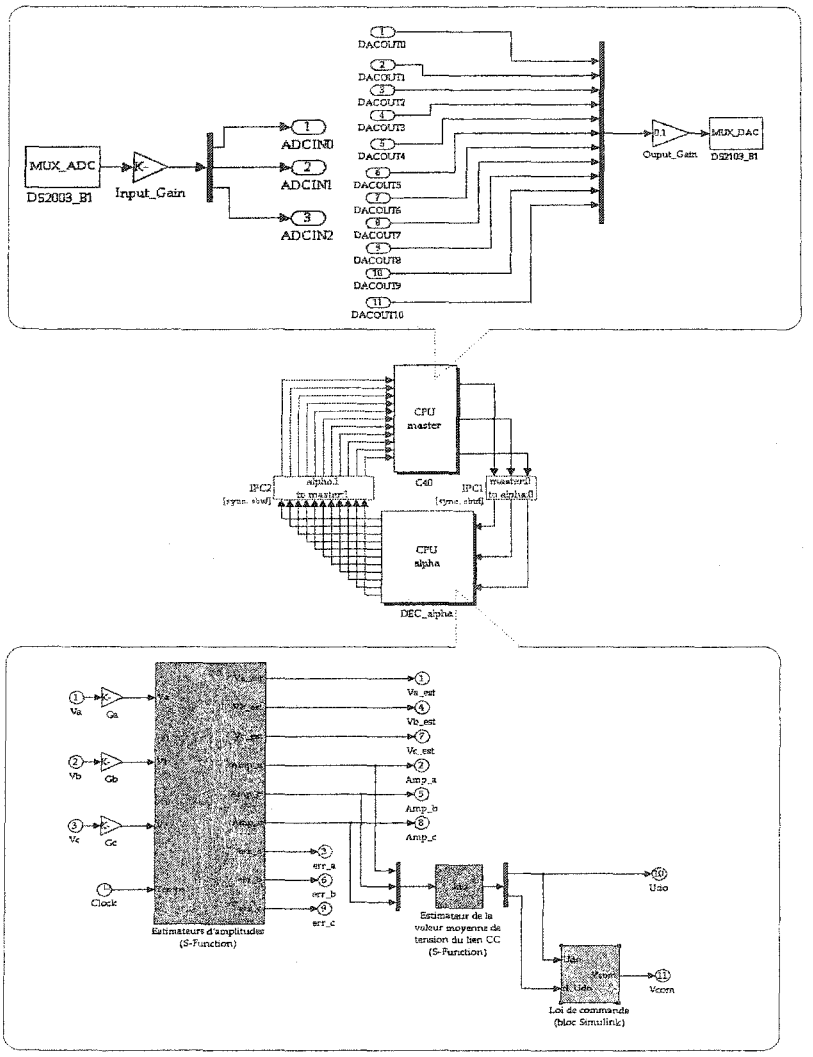


Figure 5.2 : Implémentation sous dSPACE

Dans le cadre de ce travail, la carte DS1003 est dédiée aux opérations d'entrées/sorties et la carte DS1004 est consacrée aux traitements (calculs).

L'implantation des estimateurs passe par l'écriture des S-Function en C. Une S-Function est un langage de description de bloc Simulink. Elle peut être écrite en langage Matlab, en C, en C++, en Ada ou en Fortran. Une S-function permet de créer soi-même des blocs Simulink. Ce programme est ensuite compilé en utilisant la commande *mex*.

Pour la détection des creux de tension, trois estimateurs distincts sont implémentés dans le bloc *Estimateurs d'amplitudes*. Il s'agit du même code (algorithme) à la différence que les noms qui leur sont attribués sont différents ainsi que les noms des variables statiques du programme (voir figure 5.3).

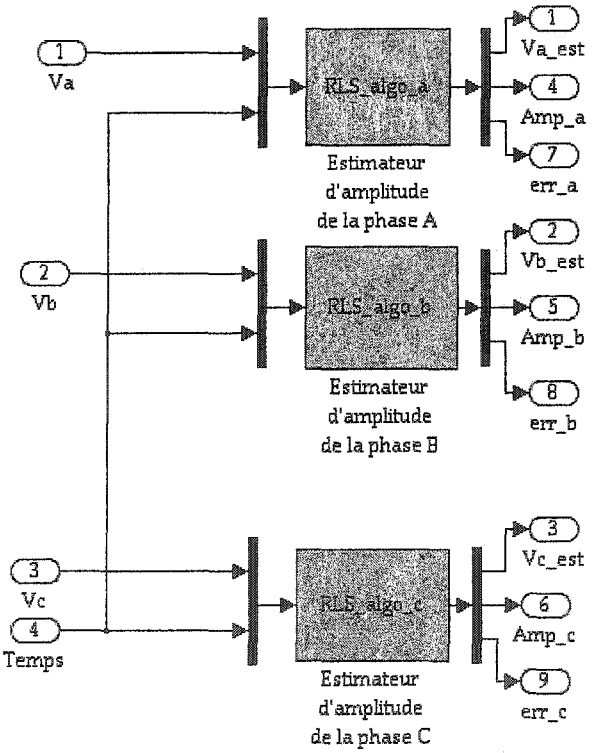


Figure 5.3 : Estimateurs d'amplitudes

Le deuxième estimateur implémenté est celui de la valeur moyenne de la tension du lien cc du redresseur à diodes. Cet algorithme a été développé sur la base de la théorie des fonctions de commutation présentée au chapitre 2. Le code de ces S-Function est présenté à l'annexe C.

La loi de compensation présentée au chapitre 4, est implantée à l'aide de blocs Simulink. Le schéma Simulink de la loi de compensation est présenté à la figure 5.4.

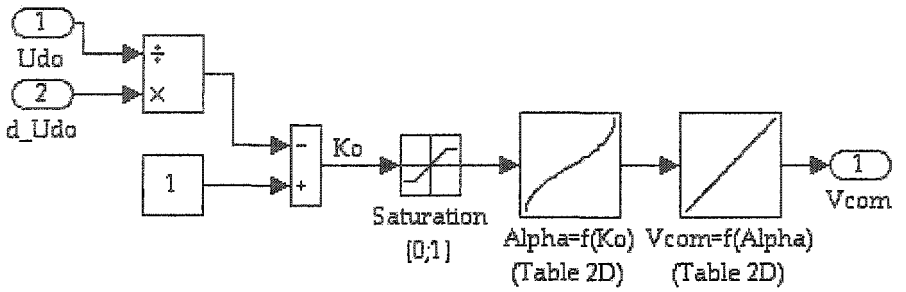


Figure 5.4 : Loi de compensation

Le bloc *Alpha = f(Ko)* est une table à deux dimensions où est implantée l'inverse de la fonction suivante pour $0 \leq K_o \leq 1$:

$$\begin{cases} K_o = 1 - \dfrac{1}{\sqrt{3}K_{T1}}\cos\alpha & si \quad 0 \leq \alpha < \dfrac{\pi}{6} \\ K_o = 1 - \dfrac{1}{\sqrt{3}K_{T1}}\left[1+\sin\left(\dfrac{\pi}{6}-\alpha\right)\right] & si \quad \dfrac{\pi}{6} < \alpha < \dfrac{2\pi}{3} \end{cases}$$

$$\text{avec } K_o = \frac{(U_{do} - \Delta U_{do})}{U_{do}}$$

ΔU_{do} est la chute de la valeur moyenne de la tension du redresseur principal. ΔU_{do} est aussi la consigne car selon la logique de compensation, la tension de commande *Vcom* qui sera appliquée au redresseur à thyristors, imposera à la sortie du redresseur auxiliaire, une tension dont la valeur moyenne U_{d1o} égale à ΔU_{do}.

Le dernier bloc c'est-à-dire *Vcom = f(Alpha)* permet de convertir l'angle d'amorçage en degré en une tension (variant de 0 à 3.3V) qui sera ensuite appliquée à la carte de commande du redresseur à thyristors.

5.3 Conception du redresseur triphasé commandé à thyristors

Il s'agit d'un pont triphasé à thyristors standard avec une diode de roue libre en sortie. Son schéma est présenté à la figure 5.5.

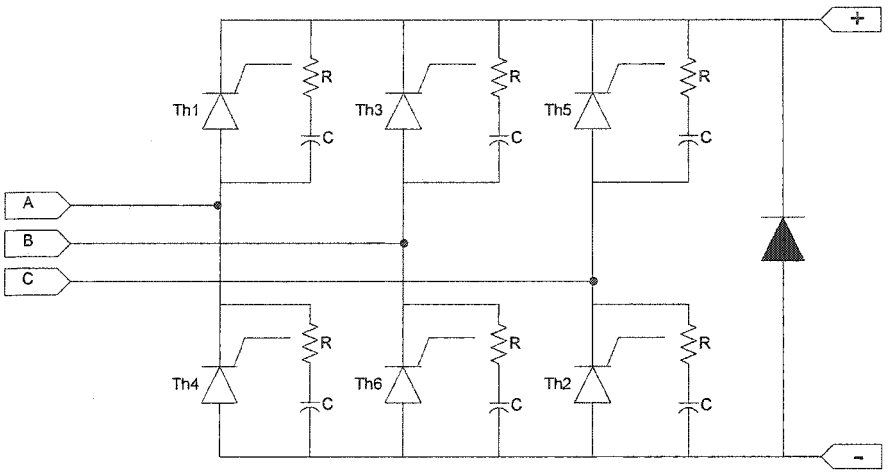


Figure 5.5 : Redresseur triphasé à thyristors avec diode de roue libre

La puissance requise est $P = U_{do\alpha} \cdot I_d = 12.5kW$. La valeur efficace de la tension d'alimentation est de 120V.

Pour un angle d'amorçage $\alpha = 0^o$, $U_{do\alpha} = 280.7V$ soit un courant $I_d = 44.53A$.

Le thyristor choisi est de la série 50RIA de International Rectifier (IR). L'analyse de sa fiche technique indique un courant efficace de 50 A et un temps de désamorçage de 110µs. Il faut souligner qu'il existe des thyristors plus rapides que ceux qui ont été choisis dans le cadre de ce travail. On peut citer l'exemple de la série ST083S de IR qui présente un temps de désamorçage variant de 10µs à 20µs et un courant efficace de 85A. Du point de vue économique, en considérant les mêmes gammes de puissances, les thyristors rapides seront quatre fois (et même plus dans certains cas) plus chers que les thyristors standards.

La diode de roue libre a les mêmes caractéristiques que la diode du module IGBT SKM 400GB125D (Semikron). Il s'agit d'une diode ultra rapide dont le courant nominal est de 390A.

Des amortisseurs (réseau RC) sont placés en parallèle sur chaque thyristor pour les protéger contre les surtensions (dV/dt). Les valeurs de la résistance R et du condensateur C ont été calculées selon la procédure décrite de la page 482 à 484 de [28].

La valeur dV/dt est limitée à 500V/μs et le facteur d'amortissement ε fixé à 0.7 nous permettant d'obtenir respectivement pour la résistance R et le condensateur C des valeurs de 3.9Ω et 0.25μF.

5.4 Conception du circuit de commande du redresseur commandé

Cette section présente la conception du circuit de commande du pont à thyristors. Les méthodes de commande des convertisseurs ca/cc peuvent se classer en deux catégories : les commandes individuelles et les commandes équidistantes [6].

Les commandes individuelles emploient trois ou six circuits de génération d'impulsions d'allumage [6]. Les impulsions d'allumage des thyristors sont référencées par rapport au passage par zéro des tensions de ligne v_{ac}, v_{ba}, v_{cb}, v_{ca}, v_{ab} et v_{bc}.

Dans les commandes équidistantes, un circuit génère une impulsion d'allumage référencée par rapport au passage par zéro d'une tension de ligne. Les autres impulsions sont produites à partir de la première dans des intervalles de temps équidistants selon la période du réseau. Tous les thyristors sont donc allumés dans des intervalles de temps égaux en régime permanent.

Notre choix se porte sur une commande individuelle; c'est-à-dire que les trois tensions de ligne (v_{ac}, v_{ba} et v_{cb}) sont prises en compte comme signaux de

synchronisation (signaux de référence) pour générer les impulsions de gâchettes des six thyristors. Cette commande a été retenue par rapport à la commande équidistante parce qu'en régime déséquilibré, il est plus facile de prédire la tension moyenne pour un angle donné. Cette assertion peut se justifier par le fait que les performances des redresseurs sont tributaires de la méthode d'allumage employée [12]. Cette carte de commande est bâtie autour d'un DSP, soit le TMS320LF2407A de Texas Instrument. Les impulsions de gâchette ont un patron prédéfini (d'après l'analyse des signaux obtenus avec le logiciel PSIM). La durée des trains d'impulsions sera de 90 degrés. Une tension qui varie de 0 à 3.3V jouera le rôle de signal de commande pour imposer l'angle d'amorçage. Le schéma synoptique de l'ensemble est présenté à la figure 5.6.

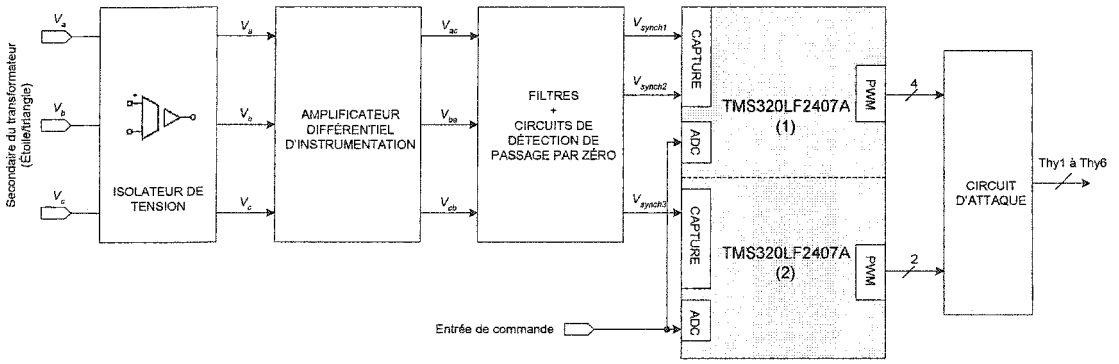


Figure 5.6 : Synoptique du circuit de commande du redresseur à thyristors

Les tensions de ligne mesurées pour la synchronisation sont filtrées. Le type de convertisseur que nous allons commander constitue un type de charge non-linéaire. Par conséquent, la tension peut être déformée due aux harmoniques générés par ce type de charge. Les tensions filtrées passent ensuite à travers des circuits de détection de passage par zéro. Les signaux ainsi mis en forme permettront au DSP à l'aide des entrées de capture de générer des impulsions synchronisées par rapport au réseau

tout en tenant compte de l'angle d'amorçage (délai) imposé par l'entrée de commande.

Les impulsions générées par le DSP sont utilisées par le circuit d'attaque pour produire des trains d'impulsions nécessaires à l'amorçage des thyristors.

Dans les prochaines sections nous présenterons la conception des différentes parties menant à la réalisation de la carte de commande.

5.4.1 Mesure des tensions de la source

Dans le cadre de ce travail, deux ensembles de mesures sont effectués. Le premier ensemble sert à l'estimation (en temps réel) des amplitudes des trois tensions d'alimentation (au primaire du transformateur T_1 de la figure 5.1) et le deuxième ensemble nous permettra d'obtenir les signaux de synchronisation pour la commande du redresseur à thyristors (au secondaire du transformateur T_1 de la figure 5.1). La conception des deux circuits étant la même, nous présentons celle du deuxième ensemble. Lors de la mesure, les tensions seront réduites à l'aide de diviseurs de tension à des niveaux acceptables par rapport aux circuits subséquents. Des isolateurs de tension sont ensuite utilisés pour fournir une isolation galvanique entre le réseau (forte puissance) et les circuits subséquents qui sont de faible puissance (cartes DSP, carte d'acquisition du système dSPACE, etc.). L'isolateur de tension choisi est ISO124P. Il est alimenté via un convertisseur cc/cc isolé : le DCP020515DP prend en entrée une tension de +5V et délivre une tension de sortie isolée de l'entrée de ±15V. La figure 5.7 présente le schéma du montage.

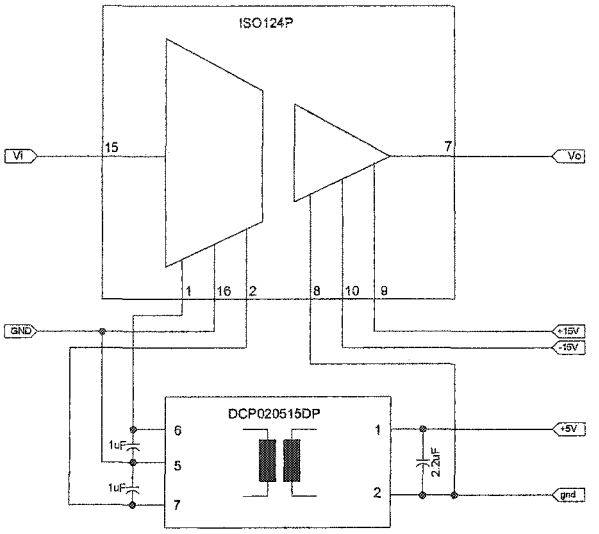


Figure 5.7 : Schéma de l'isolateur de tension

Assumant que les tensions sont déformées à cause du type de charge connecté à la source, nous utilisons des filtres passe-bande centrés sur une fréquence de 60Hz car il a comme avantage de ne pas introduire de déphasage (ou très négligeable) pour cette fréquence. On pourrait utiliser un filtre passe-bas. Cependant ce filtre a pour particularité d'introduire un déphasage entre le signal d'entrée et le signal de sortie comme le montrent les figures 5.8 et 5.9. Un déphasage introduit sur les tensions mesurées (v_{an}, v_{bn} et v_{cn}) et les tensions de synchronisation (v_{ac}, v_{ba} et v_{cb}) pourrait entraîner un délai indésirable dans la détection et la compensation des creux de tension.

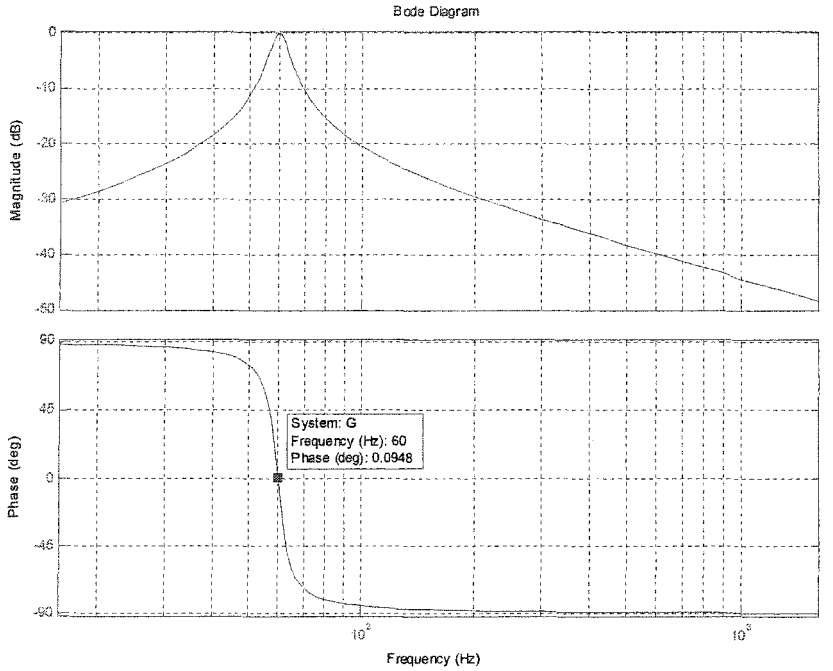


Figure 5.8 : Diagramme de Bode d'un filtre passe-bande

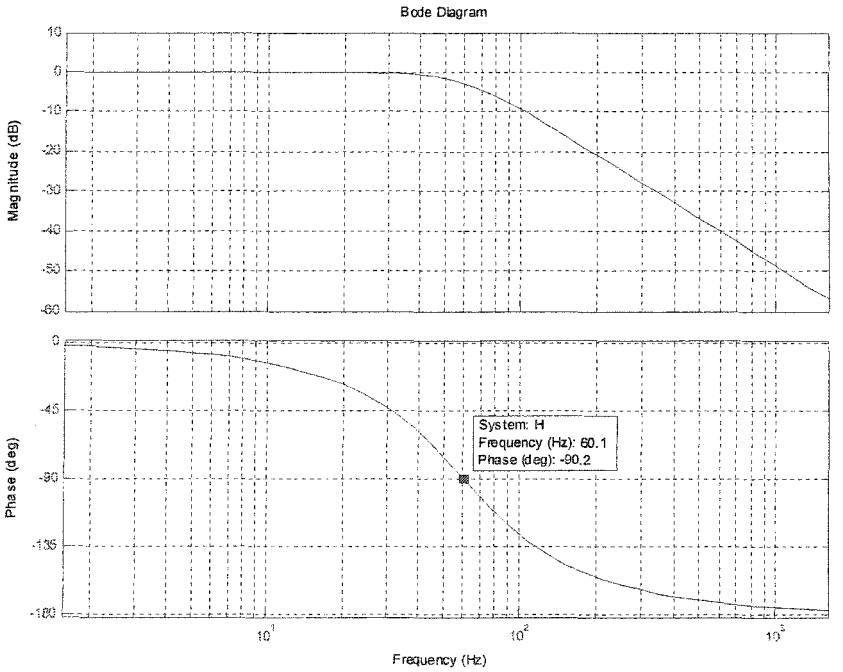


Figure 5.9 : Diagramme de Bode d'un filtre passe-bas

Le filtre passe-bande est réalisé autour du circuit UAF42 (voir figure 5.10).

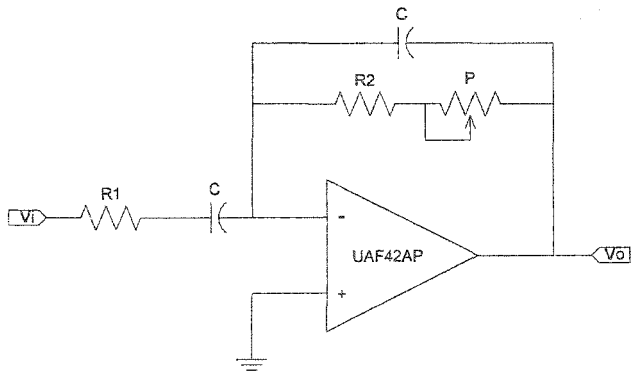


Figure 5.10 : Filtre passe-bande 2e ordre

$R_1 = 33k\Omega$, $R_2 = 22k\Omega$, $P = 10k\Omega$ et $C=100nF$;

Le potentiomètre P permet d'ajuster la bande passante B du filtre.

$B = f_2 - f_1$;

Avec $f_1 = \frac{1}{2\pi R_1 C}$ et $f_2 = \frac{1}{2\pi (R_2 + P) C}$

Comme le montrent les courbes expérimentales de la figure 5.11, l'isolateur que nous avons choisi ainsi que le filtre passe-bande que nous avons conçu introduisent un déphasage quasi-nul. Lors de l'expérimentation, dotés d'un oscilloscope de grande précision, nous avons pu déceler un déphasage variant de 30µs à 90µs (soit de 0.7° à 2°).

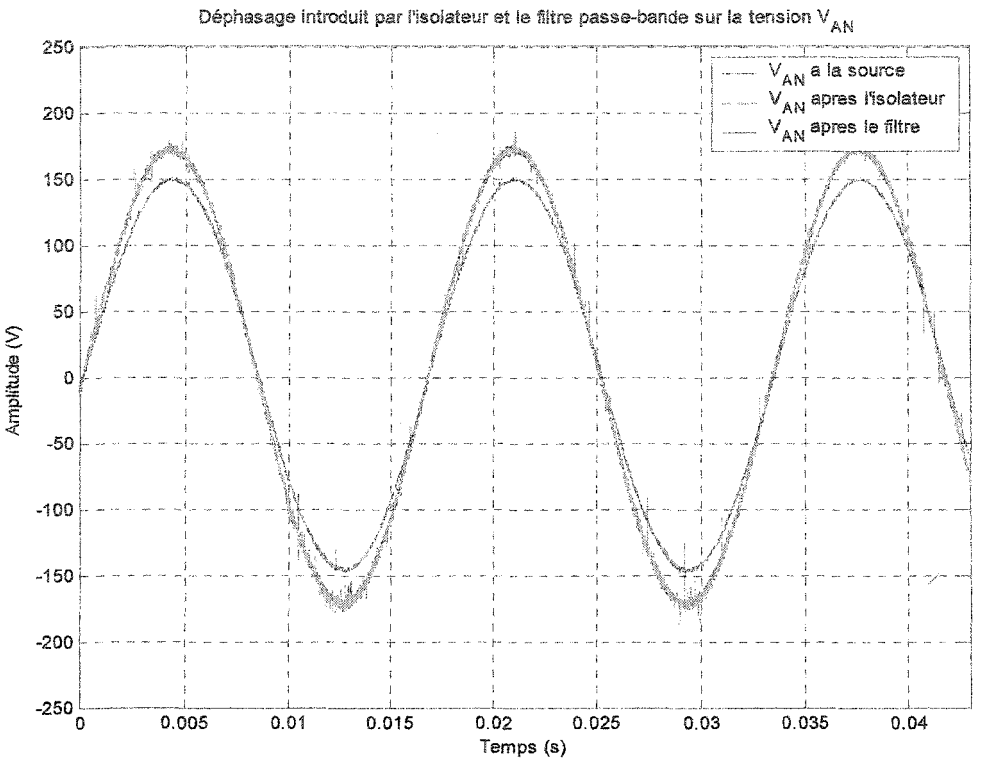


Figure 5.11 : Courbes expérimentales de la tension *Va*

5.4.2 Mise en forme des signaux de synchronisation

Ce circuit permet la mise en forme des tensions de ligne nécessaires pour la synchronisation des signaux d'amorçage. Il s'agit d'un comparateur à hystérésis bâti autour du LM311N (voir figure 5.12). L'hystérésis procure à notre circuit une certaine immunité au bruit et empêche le bruit de provoquer de faux basculements. Le point central de basculement V_{cen} est zéro.

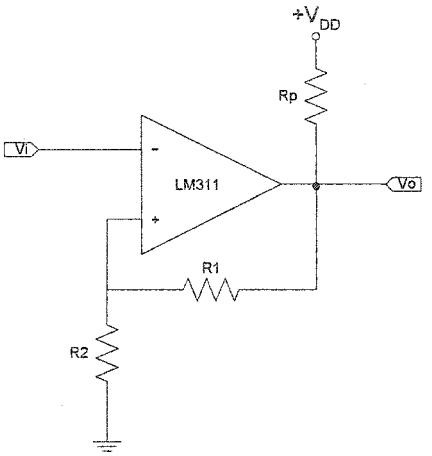


Figure 5.12 : Circuit de détection de passage par zéro

Les résistances R_1 et R_2 nous permettent de définir la largeur de l'hystérésis :

$$V_H = PSB - PIB$$

PSB : point supérieur de basculement

PIB : point inférieur de basculement

Nous imposerons $V_H = 20mV$

$$PSB = +BV_{sat} \tag{5.1}$$

$$PIB = -BV_{sat} \tag{5.2}$$

$$B = \frac{R_2}{R_1 + R_2}$$

$$(5.1) \quad \Rightarrow R_1 = \frac{(1-K)R_2}{K} \tag{5.3}$$

avec $K = \frac{PSB}{V_{sat}}$

$$(5.3) \quad \Rightarrow R_1 = 1499\, R_2$$

En posant $R_2 = 1k\Omega$, on obtient $R_1 = 1.5M\Omega$.

La V_{DD} sera égale +5V afin que le signal de sortie soit compatible avec le DSP. La résistance de « pull-up » R_P est d'une valeur de 2.7kΩ (voir figure 5.12). La figure 5.13 présente l'ensemble du circuit fournissant les signaux de synchronisation[1].

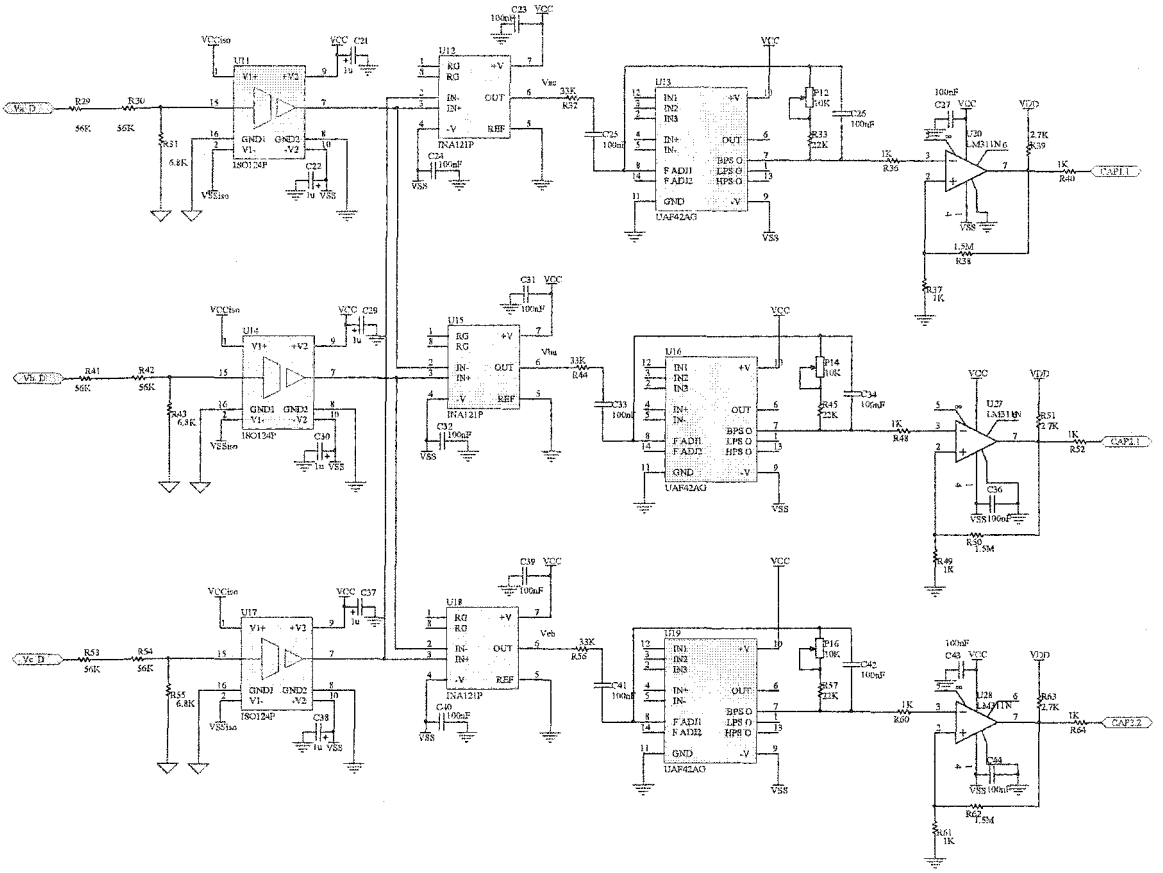


Figure 5.13 : Circuit de génération des signaux de synchronisation

La figure 5.14 présente la tension v_{ac} utilisée pour la synchronisation ainsi que le signal mis en forme présenté à l'entrée de capture du DSP.

[1] *VCCiso et VSSiso sont les tensions isolées +15V et -15V fournit par le DCP020515DP qui n'est représenté dans ce schéma*

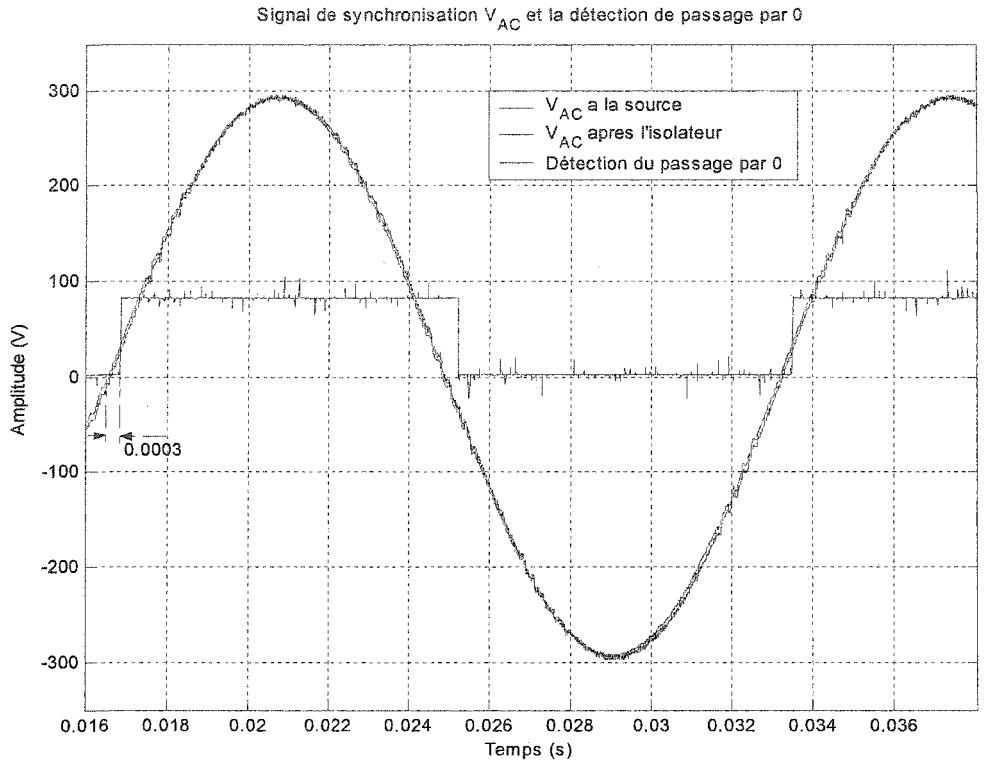


Figure 5.14 : Tension de synchronisation versus le signal mis en forme

On observe un délai de 300µs. Ce délai correspond à un angle de 6.48°. Ce délai n'a aucune incidence majeure sur la commande en terme de temps de réponse. À cause de ce délai, le pont à thyristors ne pourra être amorcé à un angle inférieur 6.48°. En réalité, la tension de commande n'ira jamais en deçà de 10°.

5.4.3 Génération des signaux de commande et circuit d'attaque

L'analyse des signaux obtenus lors des simulations du circuit de commande présenté au chapitre précédent à la section 4.4, nous a permis de définir un *patron* que nous avons ensuite implanté dans deux DSP. La figure 5.15 présente le principe de la génération des signaux d'amorçage pour les thyristors 1 et 4 tel qu'implanté dans le logiciel PSIM .

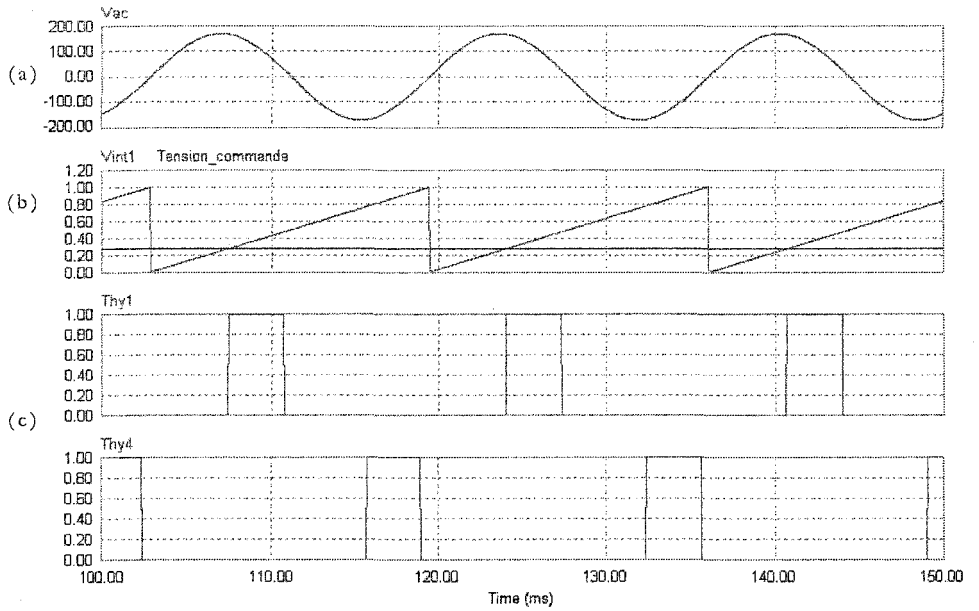


Figure 5.15 : Principe de la génération des signaux d'amorçage des thyristors 1 et 4

a) Premier signal de synchronisation
b) Signal intégré et signal de commande
c) Thy1, Thy4 : Signaux de gâchette des thyristors 1 et 4

Une fois les impulsions Thy1 et Thy4 générées par le DSP, elles sont envoyées à des circuits d'attaque qui permettront de fournir le courant de gâchette nécessaire à l'amorçage des thyristors 1 et 4. Dans les sous-sections qui suivent, nous présenteront le patron sur lequel se base la génération des signaux et l'organigramme du programme DSP. Le circuit d'attaque, ainsi que son dimensionnement seront présentés.

a) Génération des signaux

Les impulsions d'amorçage {thy2, thy5} et {thy1, thy4, thy3, thy6} sont générées selon les patrons présentés respectivement aux figures 5.16 et 5.17.

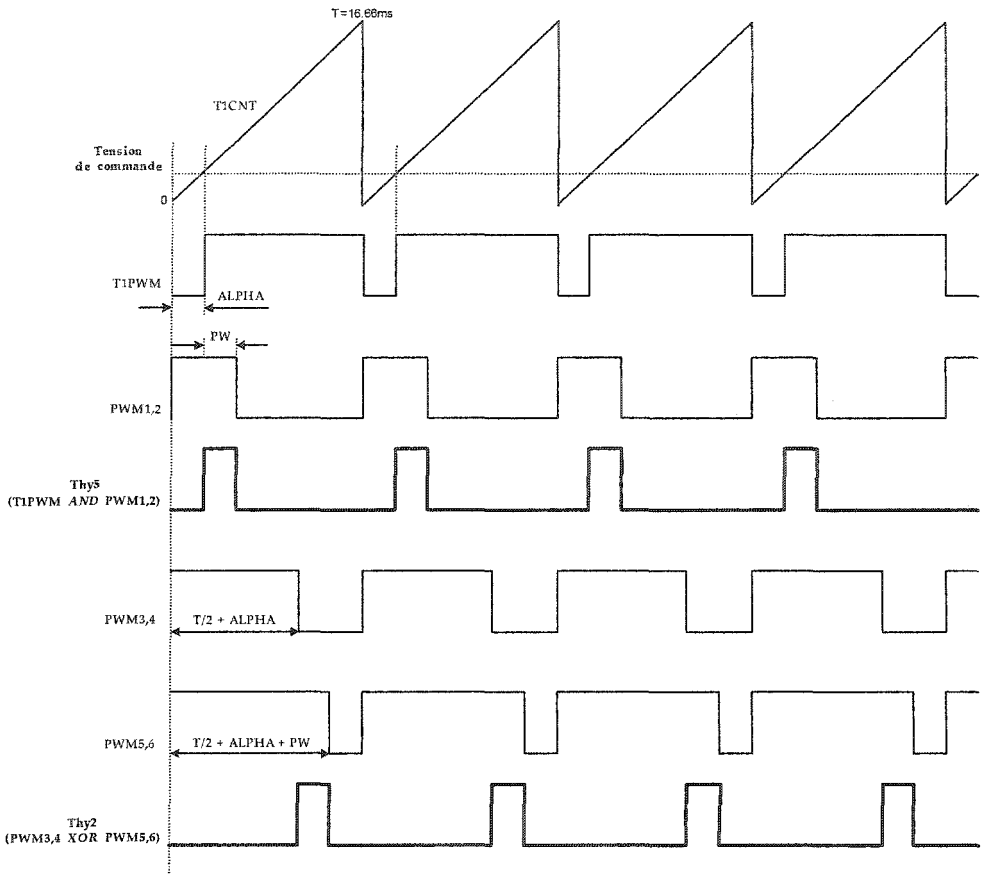


Figure 5.16 : Patron des signaux générés par le deuxième DSP

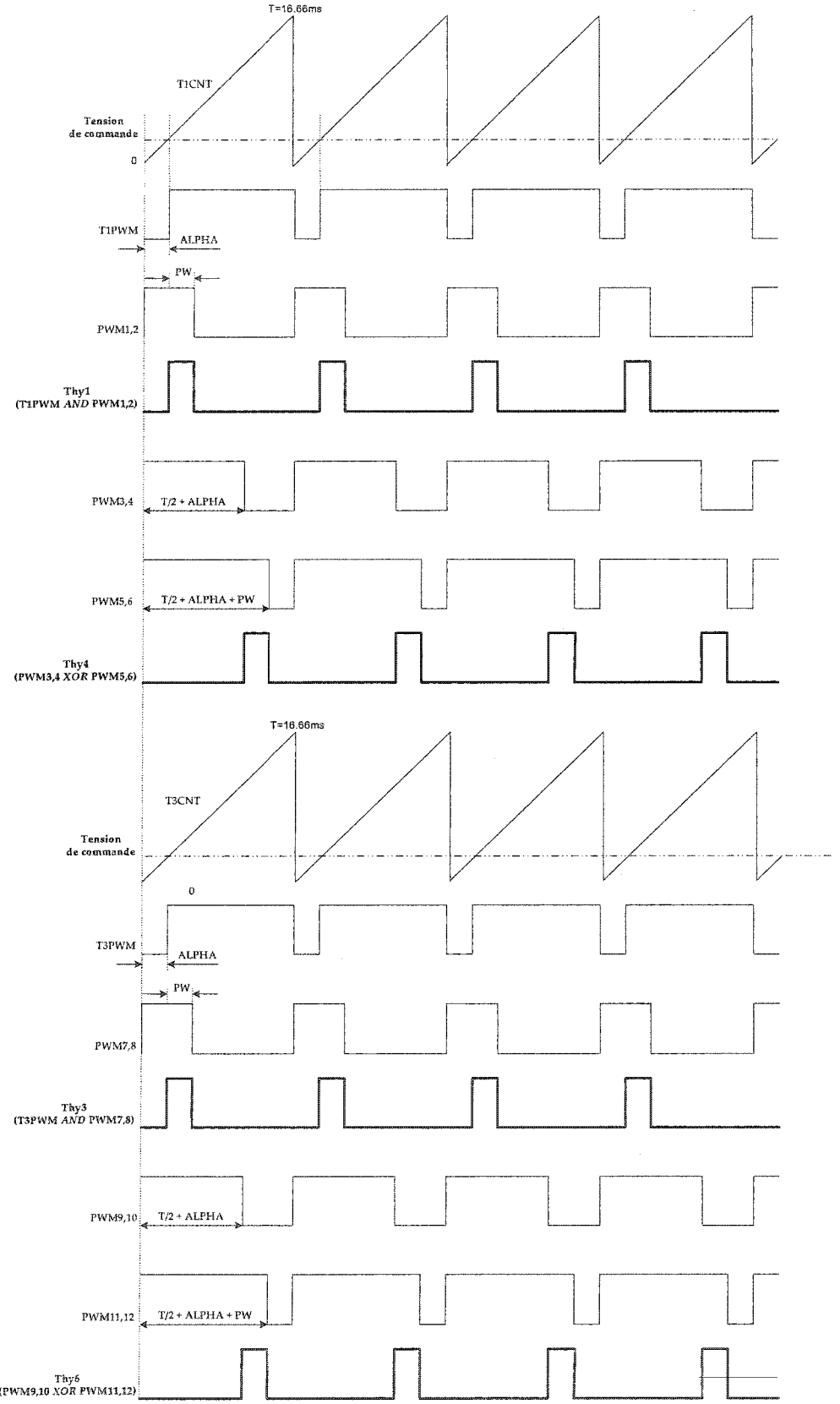


Figure 5.17 : Patron des signaux générés par le premier DSP

b) Organigramme des programmes DSP

Les programmes principaux sont représentés aux figures 5.18 et 5.19.

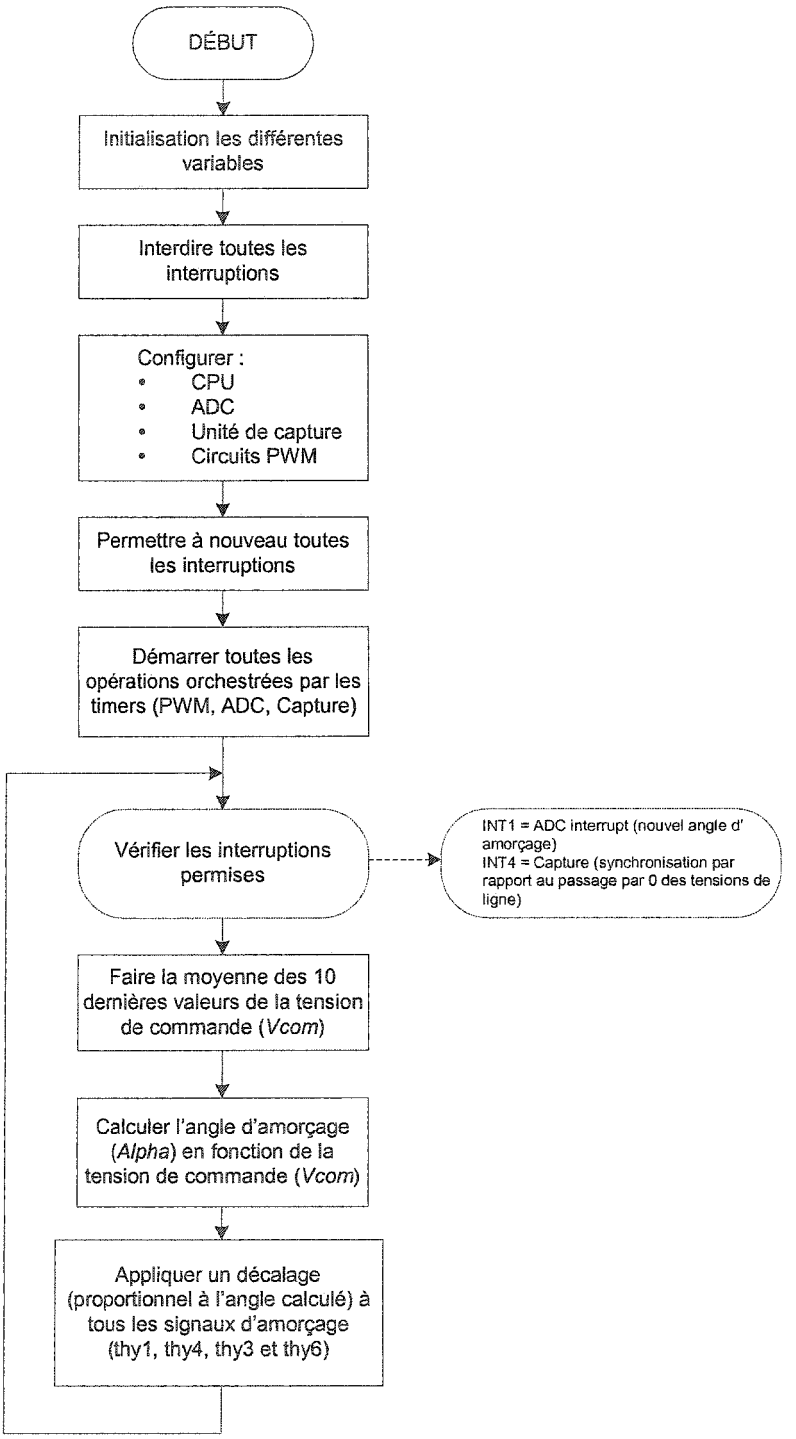


Figure 5.18 : Organigramme du programme principal du 1er DSP

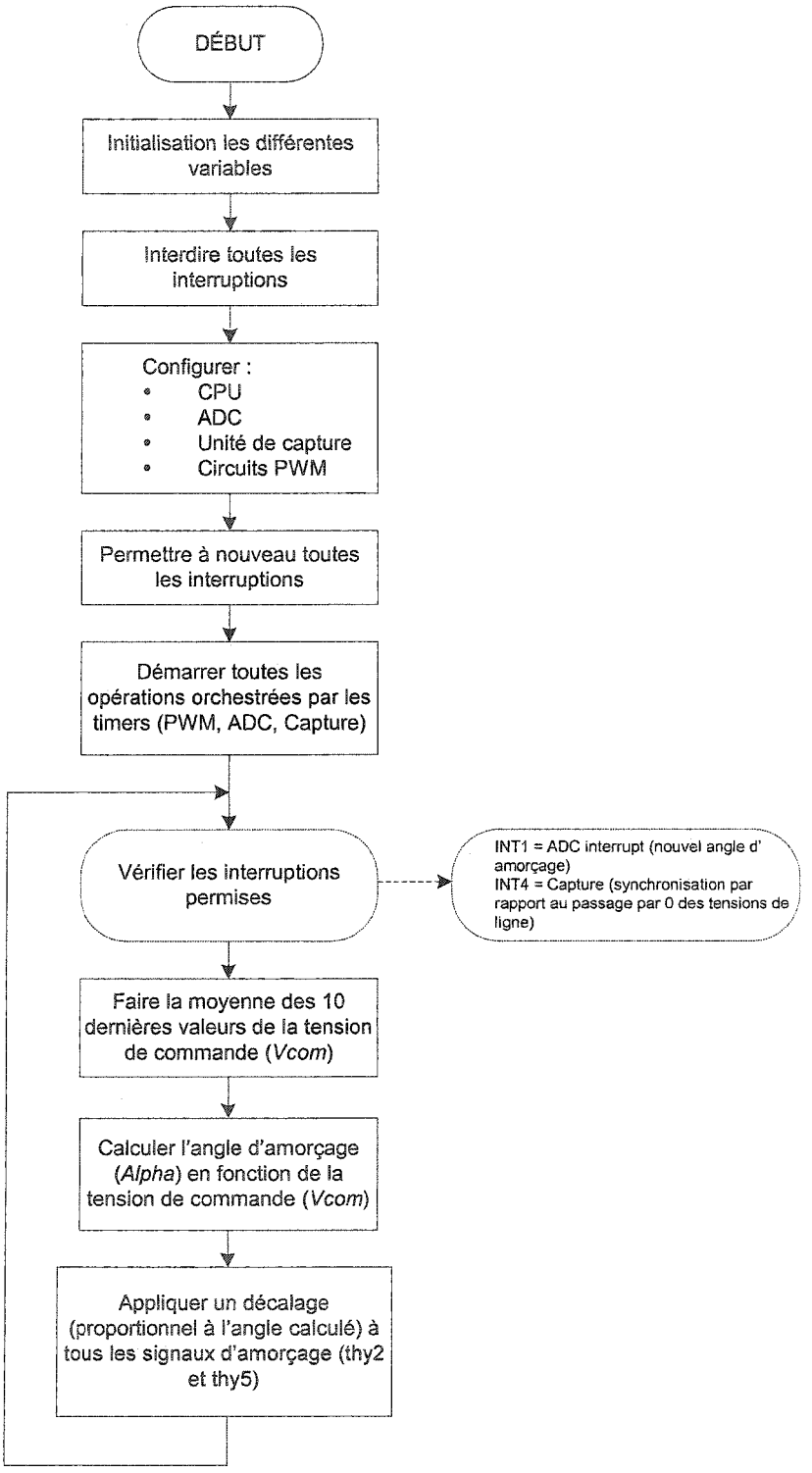


Figure 5.19 : Organigramme du programme principal du 2e DSP

Dans les deux programmes principaux, la sous-routine d'interruption pour la conversion analogique numérique est la même. Son organigramme est présenté à la figure 5.20.

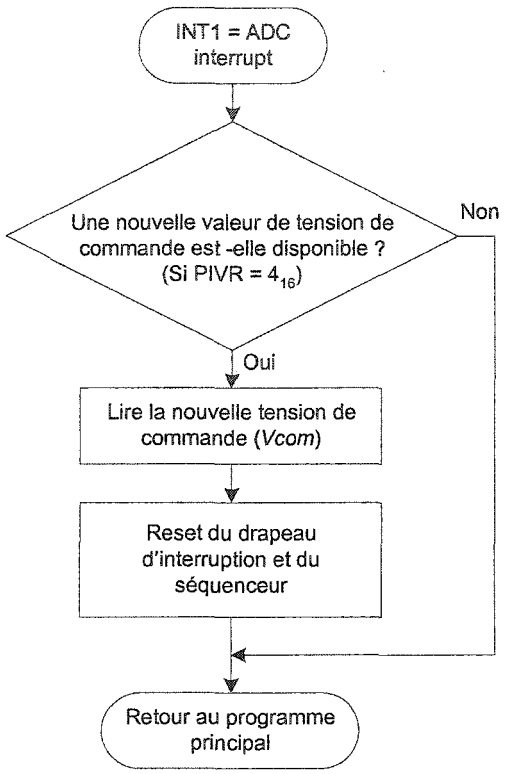


Figure 5.20 : Sous-routine de conversion analogique numérique

Ce n'est cependant pas le cas pour les sous-routines de capture car le premier DSP doit se synchroniser sur deux signaux de références. Il utilise à cet effet, deux entrées de capture (CAP1 et CAP2) tandis que le deuxième DSP utilise un seul signal de synchronisation, donc une seule entrée de capture (CAP2). Ces sous-routines sont présentées aux figures 5.21 et 5.22.

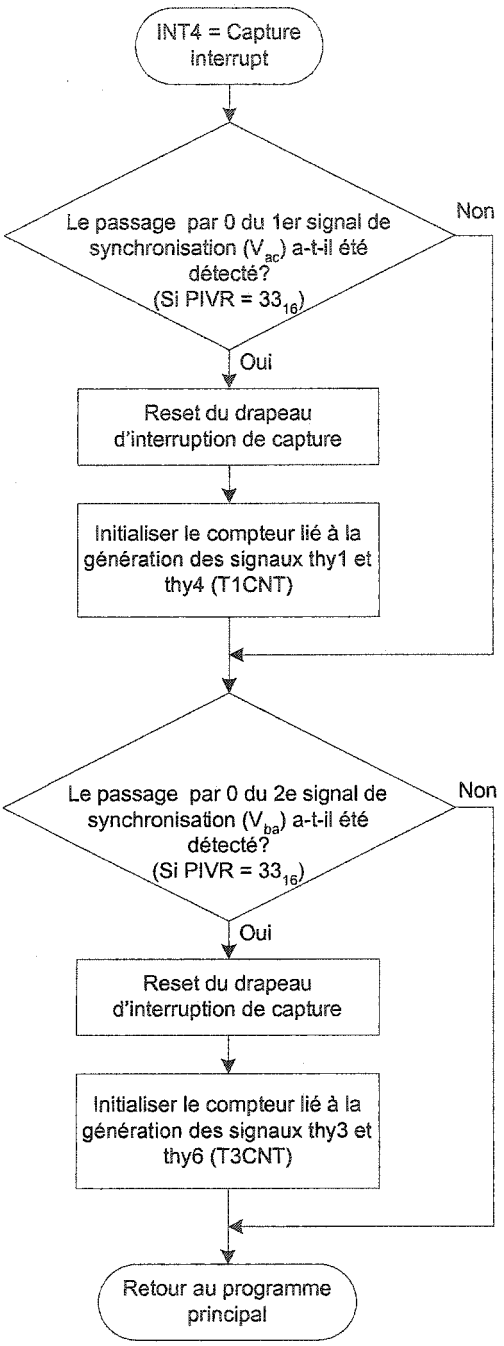


Figure 5.21 : Sous-routine de capture du 1er DSP

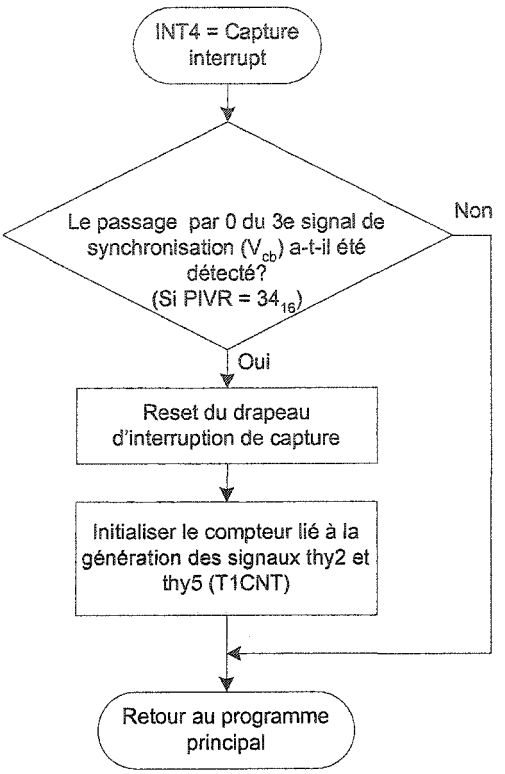


Figure 5.22 : Sous-routine de capture du 2e DSP

Les programmes DSP sont sauvegardés en permanence dans la mémoire flash du TMS320LF2407A. Les impulsions d'amorçage sont générées par les DSP aussitôt que l'alimentation est présente sur la carte. Le fonctionnement des DSP ne dépend pas d'un autre système (un autre ordinateur ou de l'émulateur comme c'est le cas lors du développement). C'est une application embarquée. La façon de réaliser ce type d'application avec le TMS320LF2407A (version eZdsp) est présentée à l'annexe D. Les codes des deux programmes sont présentés à l'annexe E.

c) Le circuit d'attaque

Ce circuit sert d'interface entre le circuit de commande (faible puissance) et le circuit de forte puissance. Il procure une isolation galvanique grâce au transformateur

connecté entre les deux circuits sus mentionnés. Le schéma du montage est celui de la figure 5.23.

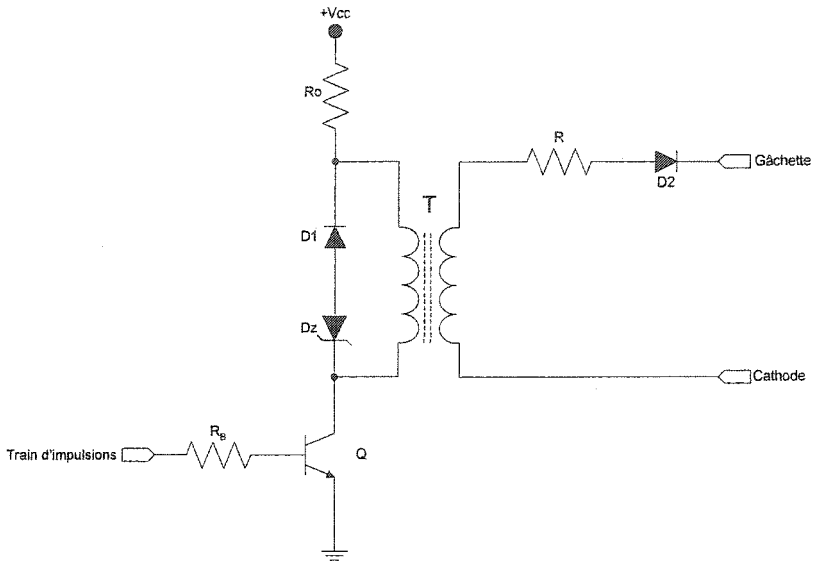


Figure 5.23 : Circuit d'attaque d'un thyristor

Pour éviter de saturer le transformateur, les impulsions d'allumage générées par le DSP seront d'abord converties en trains d'impulsions. Ces impulsions seront de largeur bien définie, respectant le «*produit V·T*» ; cette constante est donnée par le constructeur et exprimée en V·µs. La deuxième contrainte est une démagnétisation totale entre chaque cycle de fonctionnement. Les diodes D_1 et D_Z permettent de rencontrer cette dernière.

Considérons :

- Thyristors : IR50RIA
  - Tension gâchette-cathode : $V_{GT} = 2.5V$
  - Courant continu de gâchette nécessaire à l'amorçage : $I_{GT}=100mA$
- Transformateur d'impulsions: 611C (Hammond)
  - Rapport de transformation : 1 : 1
  - Inductance primaire $L_P = 56mH$

- Constante VT = 7250Vμs
- Tension d'alimentation : 15V

Détermination des résistances de gâchette et de la base

Pour amorcer les thyristors, nous assumons un courant de gâchette d'une valeur moyenne de 200mA. En adoptant un rapport cyclique de 50% du train d'impulsions, la valeur crête d'une impulsion sera 400mA.

Donc, $I_2 = 400mA$

I_2 est le courant au secondaire du transformateur T (voir figure 5.23). Le rapport de transformation étant de 1, I_2 est égal au courant du primaire du transformateur I_1.

$$R_G = \frac{Vcc - V_{D2} - V_{GT}}{I_2} \qquad (5.4)$$

$$(5.4) \Rightarrow R_G = \frac{15 - 1.0 - 2.5}{400 \times 10^{-3}} = 28.75\Omega$$

Comme valeur normalisée, nous prendrons $R_G = 27\Omega$.

La fréquence minimale du signal de commande dépend de la constante *VT* du transformateur d'impulsions. La largeur maximale d'une impulsion sans saturer le transformateur est :

$$t = \frac{VT}{Vcc} \qquad (5.5)$$

Soit $t = 483.33\mu s$;

avec un rapport cyclique de 50%, cela correspond à une fréquence minimale de 2.069kHz.

Nous imposerons une fréquence de 6.25kHz. La largeur des impulsions t devient 80µs. En calculant le produit $Vcc \times t$, nous constatons qu'il est très inférieur à la constante $V \cdot T$.

Lors de la conduction du transistor Q, le courant magnétisant est donné par :

$$I_0 = t \cdot \frac{Vcc}{L_P} \tag{5.6}$$

$$I_0 = 80 \times 10^{-6} \cdot \frac{15}{56 \times 10^{-3}} = 21.43\text{mA}$$

Le courant dans le collecteur du transistor est :

$$Ic = I_0 + I_2 \tag{5.7}$$

$$Ic = 21.43\text{mA} + 400\text{mA} = 421.43\text{mA}$$

En supposant que le gain β est égal à 100, le courant de base nécessaire est donc 4.2mA.

La résistance de base maximale vaut : $R_{Bmax} = \frac{15 - 0.7}{4.21 \times 10^{-3}} = 3.39\text{k}\Omega$

Nous prendrons $R_B = 2.4\text{k}\Omega$.

<u>Détermination de la diode Zener</u>

Pour assurer une démagnétisation complète du transformateur, on choisit une diode Zener de tension inverse supérieure à 15V. Aussi, lors du blocage, le transistor est

soumis à une tension équivalente à $Vcc + V_{D1} + V_{DZ}$. Pour éviter la destruction du transistor, cette tension équivalente doit être inférieure à la tension V_{CE}. On choisira une diode Zener de 18V, soit la diode 1N4746A.

5.5 Conception du circuit de commande du contrôle de l'injection du 3e harmonique

La structure du circuit de commande est proposée [23]. Son schéma est présenté à la figure 5.24.

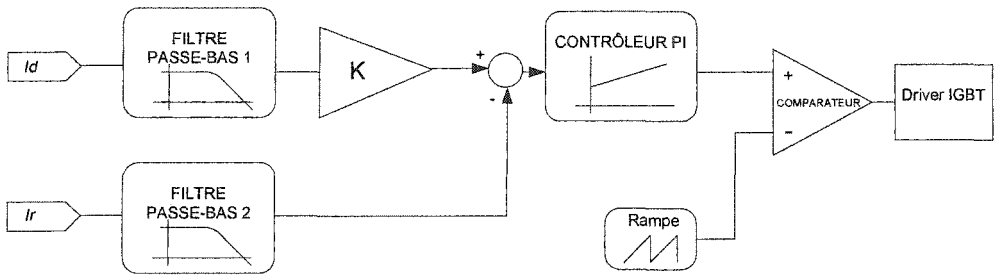


Figure 5.24 : Commande du contrôle du circuit d'injection

Filtres passe-bas et gain K

Les filtres passe-bas 1 et 2 sont respectivement d'ordre 2 et 4. Ils ont un gain unitaire et une fréquence de coupure de 70Hz. Les figures 5.25 et 5.26 représentent des filtres passe-bas Sallen-Key d'ordre 2 et 4. Ils offrent l'avantage d'utiliser des composants (R et C) identiques. Le potentiomètre R_2 permet d'ajuster le gain du filtre.

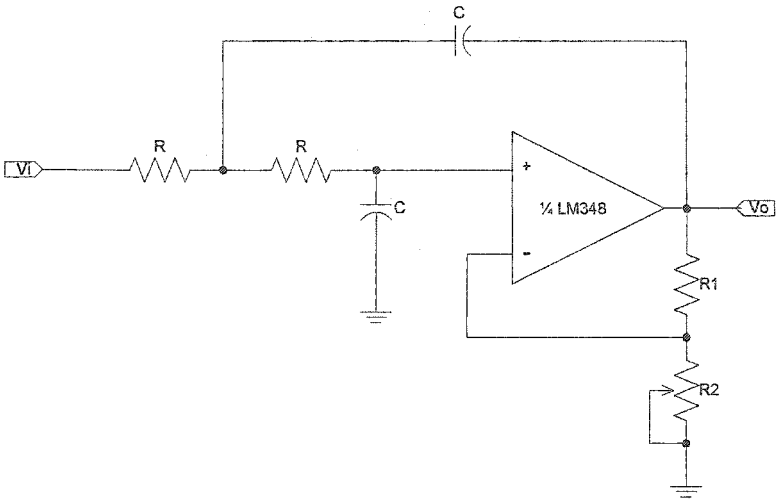


Figure 5.25 : Filtre passe-bas 1

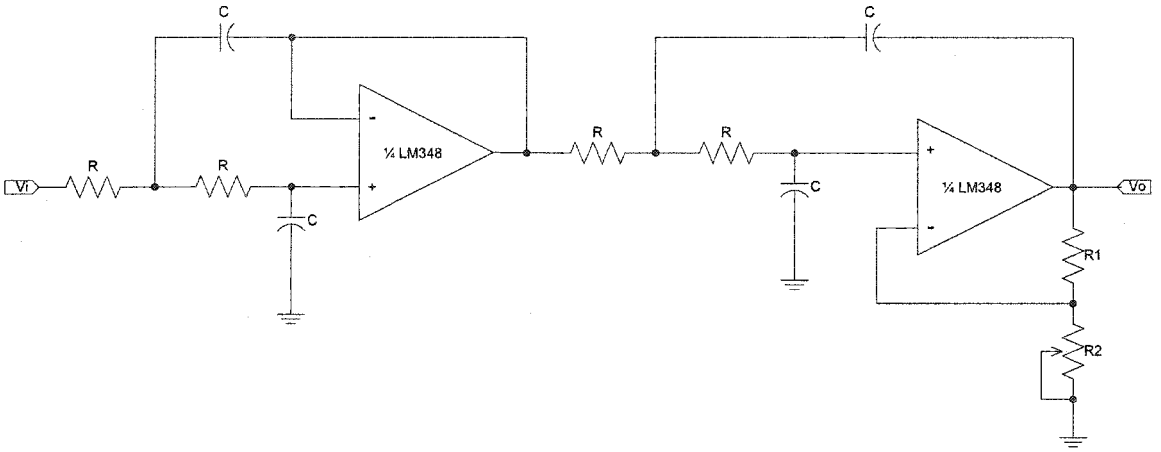


Figure 5.26 : Filtre passe-bas 2

La fréquence de coupure est [29]:

$$f_c = \frac{1}{2\pi R C} \tag{5.8}$$

En assumant que fc = 70Hz,

$$\text{(5.8) devient } RC = \frac{1}{2\pi f_c} \tag{5.9}$$

si $C = 100nF \Rightarrow R = 22.74\,k\Omega$

d'où $R = 22k\Omega$ (valeur normalisé)

$$f_{c\,réelle} = \frac{1}{2\pi \times 22000 \times 100 \times 10^{-9}} = 72.34\,Hz$$

Le gain K est l'inverseur de la figure 5.27.

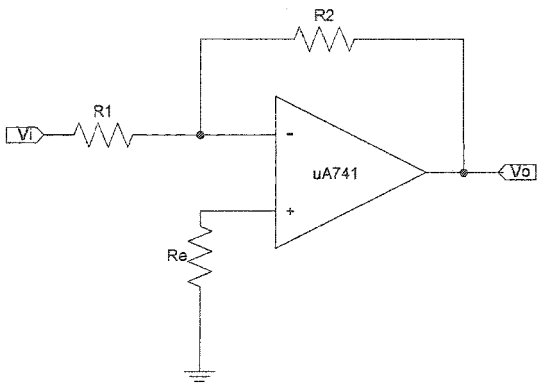


Figure 5.27 : Schéma du gain K

$$K = -\frac{R_2}{R_1} \qquad (5.10)$$

si $|K| = 0.16$ et R_1=10kΩ

donc, R_2 = 1.6kΩ. Nous remplacerons R_2 par une résistance de 1kΩ en série avec un potentiomètre de 1kΩ pour mieux ajuster le gain K.

La résistance R_e permet de rendre égaux les courants de polarisation de l'amplificateur opérationnel. Elle aura pour valeur 1.2kΩ.

Contrôleur PI

Le contrôleur utilisé est présenté à la figure 5.28. Il s'agit d'un contrôleur proportionnel intégral.

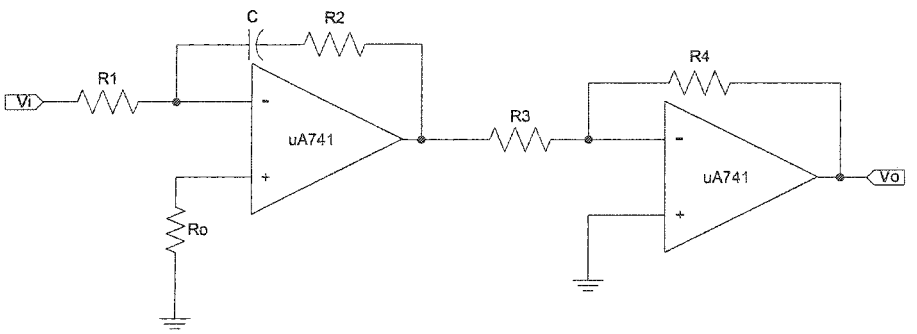


Figure 5.28 : Schéma du contrôleur PI

La fonction de transfert du contrôleur est :

$$\frac{V_o}{V_i} = A\left(K_P + \frac{K_I}{s}\right) \tag{5.11}$$

avec

$$K_P = \frac{R_2}{R_1} \tag{5.12}$$

$$K_I = \frac{1}{R_1 C} \tag{5.13}$$

$$A = \frac{R_4}{R_3} \tag{5.14}$$

posons $K_P = 2$, $K_I = 20$ *et* $A = 1$ (Ces valeurs sont proposées en [27]).

Si C = 10μF,

(5.13) devient $R_1 = \frac{1}{K_I C} = \frac{1}{20 \times 10 \times 10^{-6}} = 5k\Omega$

R_1 sera un potentiomètre de 10kΩ pour permettre d'ajuster les gains proportionnel et intégral.

(5.12) ⇒ $R_2 = 10k\Omega$.

(5.14) ⇒ $R_3 = R_4 = 2.4k\Omega$.

Comparateur et pilote pour IGBT

Le schéma du comparateur est présenté à la figure 5.29.

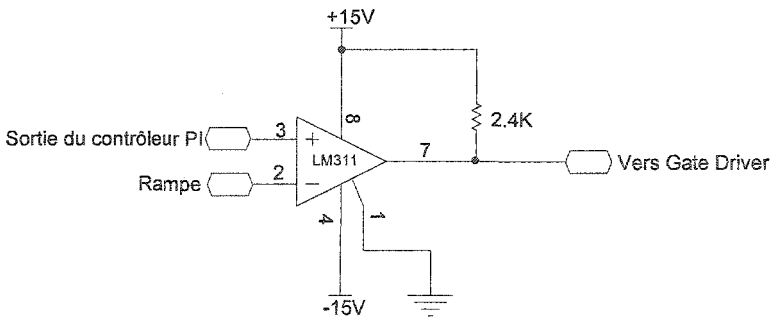


Figure 5.29 : Schéma du comparateur

Notre choix s'est porté sur le pilote (driver) M57160AL de la compagnie Powerex.

5.5 Résultats expérimentaux

Dans cette section, nous présentons quelques courbes expérimentales. Elle est divisée en trois sous-sections qui présentent respectivement l'amélioration du taux global de distorsion du courant dans la ligne, les performances de l'algorithme de détection de creux de tension et les performances du système auxiliaire de compensation en entier. Il faut noter que l'abscisse sur certaines courbes possède des valeurs négatives. Pour enregistrer les courbes des différentes tensions lors d'un creux de tension, nous avons utilisé une synchronisation automatique sur la tension de commande. L'on remarquera que l'origine de l'axe des abscisses coïncide avec la détection d'un creux de tension (voir figure 5.31). La partie négative de l'axe des abscisses correspond à un pre-amorçage de 20%; c'est-à-dire lorsqu'un creux de tension est détecté, l'oscilloscope conserve 20% des 100000 points que contient la fenêtre d'acquisition avant que le creux de tension ne se produise.

5.5.1 Amélioration du courant dans la ligne

La figure 5.30 présente le courant dans la ligne avec et sans le système auxiliaire d'injection du troisième harmonique.

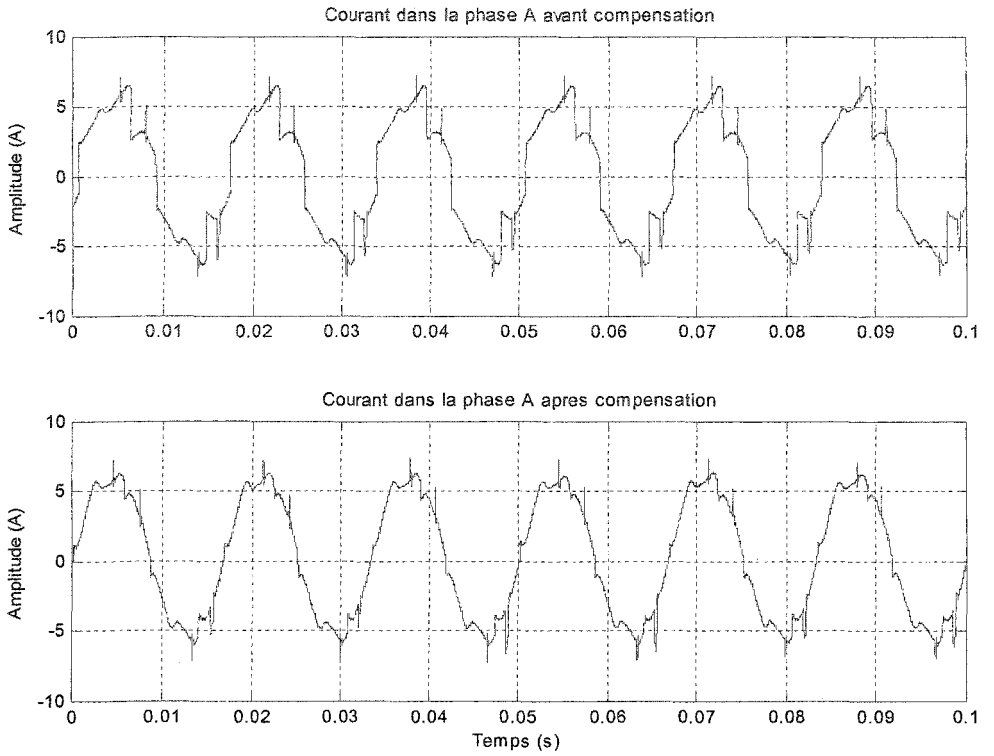


Figure 5.30 : Courant dans la ligne avec et sans le circuit d'injection du 3e harmonique

Le taux de distorsion global du courant dans la phase A avant la compensation est de 14.09% tandis qu'après la compensation, il est ramené à 6.83%. Il s'agit d'une amélioration du THD de 48.5%.

5.5.2 Performances de l'algorithme de détection de creux de tension

Une compensation fiable et efficace repose sur une détection rapide et une bonne estimation des creux de tension. La figure 5.31 présente les résultats expérimentaux de l'algorithme de détection. On remarque que les courbes de la tension mesurée et la tension estimée (reconstituée) se superposent.

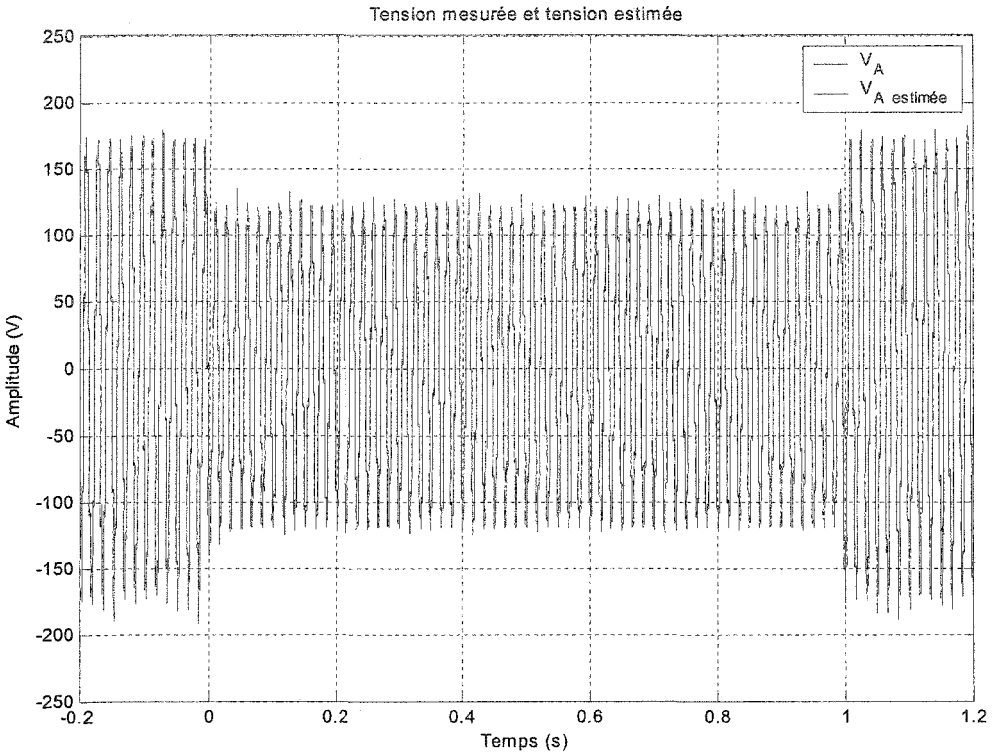


Figure 5.31 : Formes d'ondes de tensions mesurée et estimée durant un creux de tension

Trois régions de la figure 5.31 sont intéressantes à analyser : le début et la fin d'un creux de tension (régimes transitoires) et en l'absence de creux de tension (régime permanent). Les agrandissements des régions ci-haut mentionnées sont présentés de la figure 5.32 à la figure 5.37.

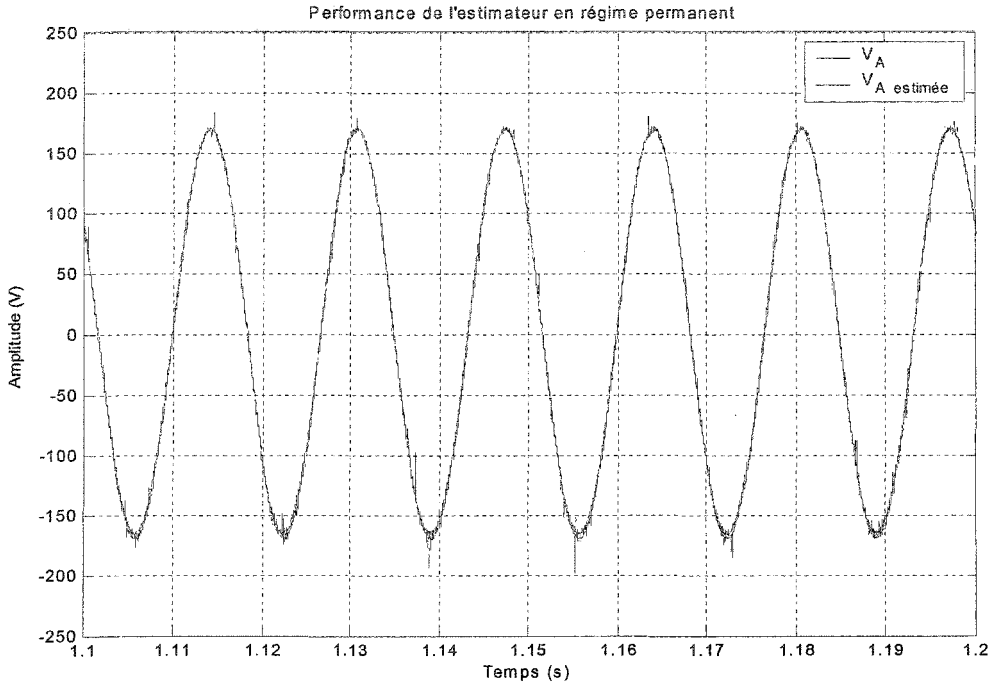


Figure 5.32 : Formes d'ondes de tensions mesurée et estimée en régime permanent

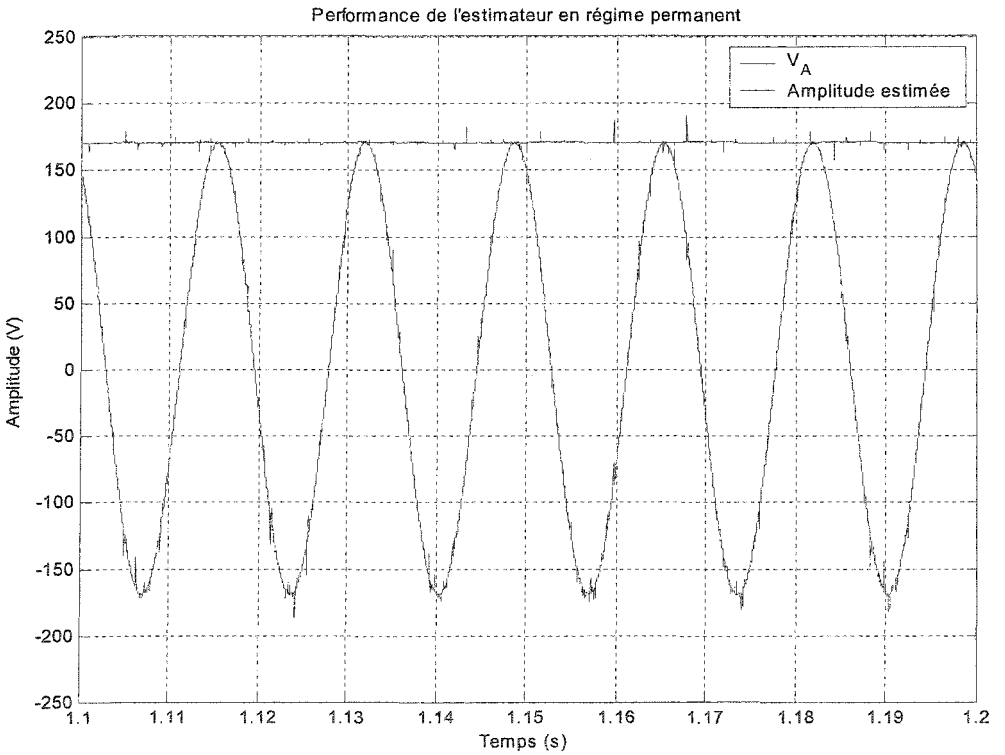


Figure 5.33 : Amplitude de tension estimée en régime permanent

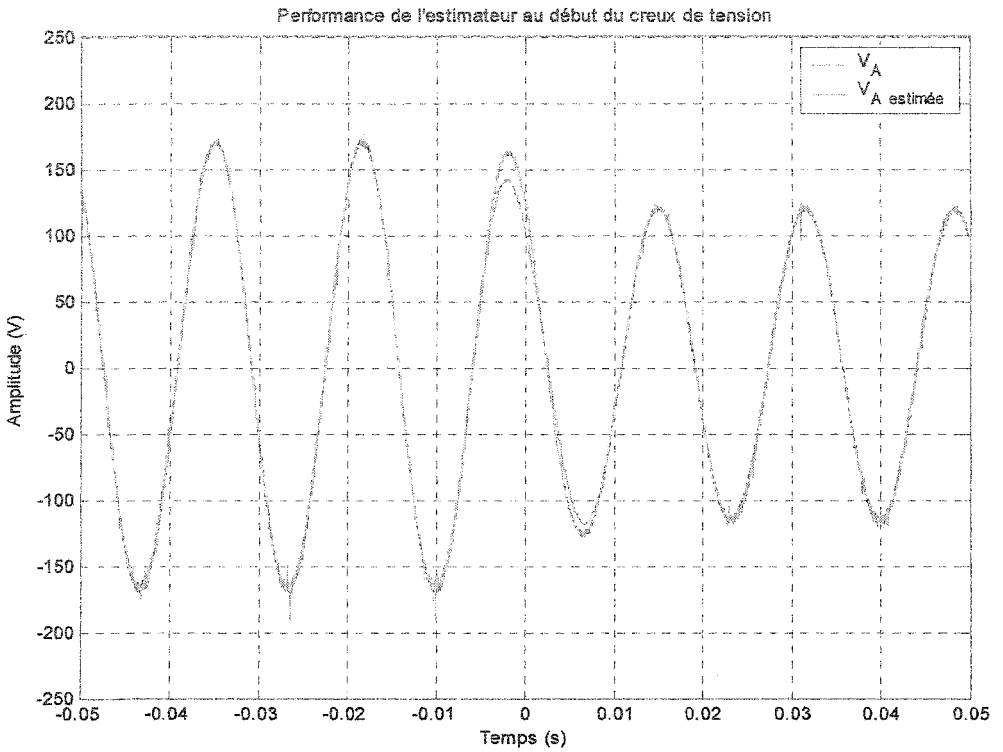


Figure 5.34 : Formes d'ondes de tensions mesurée et estimée au début d'un creux de tension

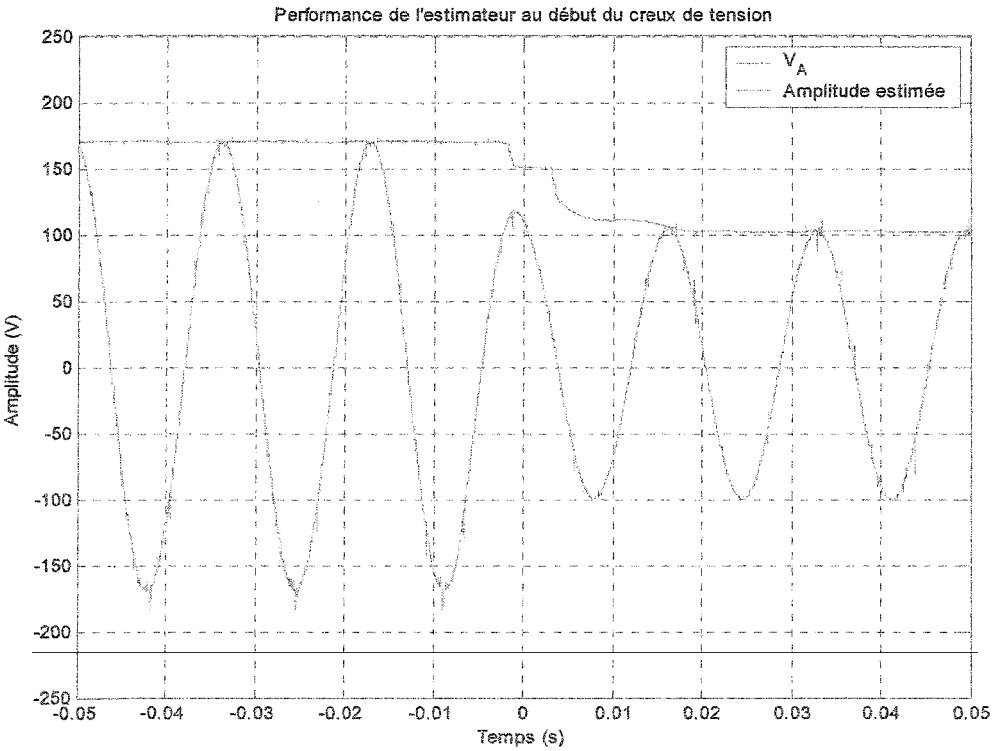


Figure 5.35 : Amplitude de tension estimée au début d'un creux de tension

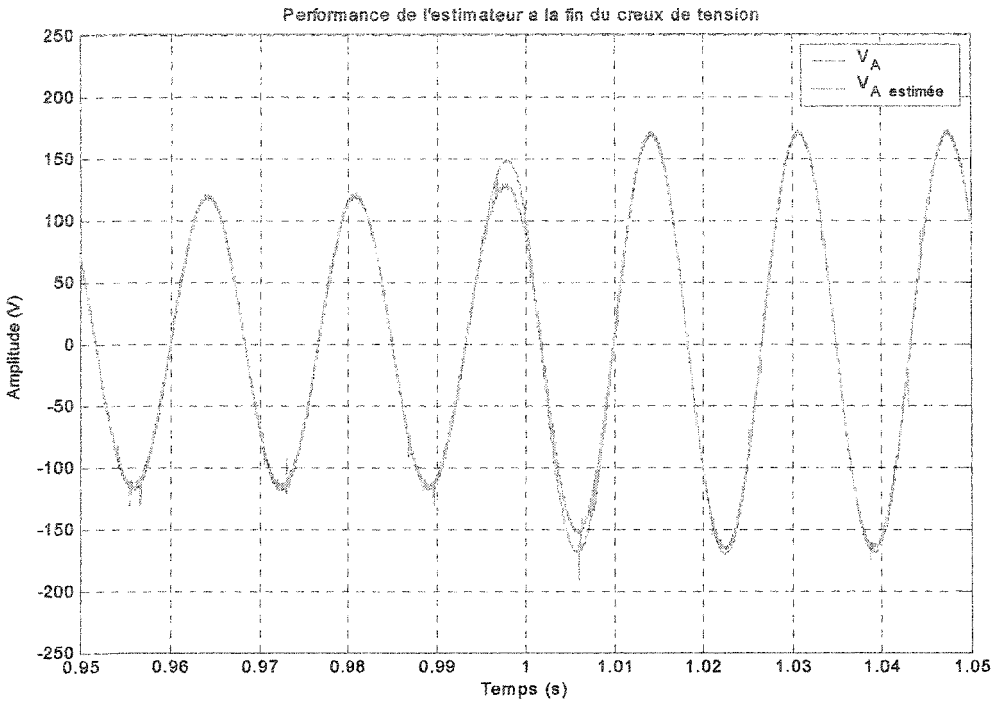


Figure 5.36 : Formes d'ondes de tensions mesurée et estimée à la fin d'un creux de tension

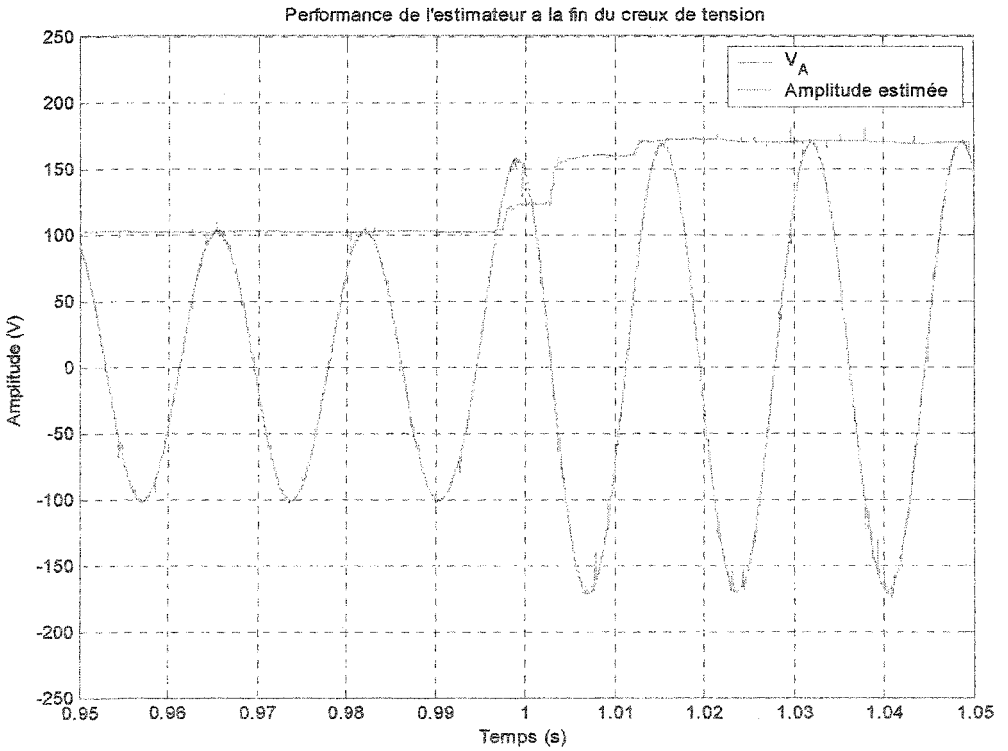


Figure 5.37 : Amplitude de tension estimée à la fin d'un creux de tension

En régime permanent, l'estimation de l'amplitude est assez bonne (voir figure 5.33). En revanche, en régime transitoire, les performances ne sont pas aussi bonnes qu'en simulation. Le tableau 5-1 compare les délais obtenus en simulation et lors de l'expérimentation. Ce délai représente l'intervalle de temps que met l'algorithme à converger vers la bonne valeur de l'amplitude.

Tableau 5-1 : Délai lors de la détection de creux de tension

	Délai lors de la détection	
Région critique	***Simulation***	***Expérimentation***
Début du creux	0.83 ms	6.8 ms
Fin du creux	4.8 ms	15.2 ms

Une raison qui peut expliquer cette différence est le pas d'échantillonnage en simulation et lors de l'expérimentation. En simulation, le pas d'échantillonnage était de 65.1µs et pendant l'expérimentation, à cause des limitations matérielles, nous n'avons pas pu implémenter les algorithmes avec un pas d'échantillonnage inférieur à 150µs.

5.5.3 Performances du système de compensation de creux de tension

Les paramètres du système sont consignés dans le tableau 5-2.

Tableau 5-2 : Tableau comparatif des paramètres utilisés en simulation et lors de l'expérimentation

Désignation	***Simulation***	***Expérimentation***
Source d'alimentation	208V_{LL} - 60 Hz	208V_{LL} - 60 Hz
Transformateur T_1	10kVA - 208V/208V	10kVA - 208V/208V
Transformateur T_2	1.5kVA - 35V/210V	1.5kVA - 35V/210V
Filtre LC	L=11.7mH, C=60µF, r=0.2Ω	L=11.7mH, C=60µF, r=0.2Ω
Inductance de lissage L_d	10mH	10mH
Condensateur C_d	1100µF	1100µF
Charge R_d	6.2Ω	60Ω

Comme en simulation, trois types de creux de tension nous ont permis de tester le système auxiliaire de compensation. Il s'agit du :

- type A : creux de tension triphasé de 30% d'une durée de 1 seconde,
- type B : creux de tension monophasé de 40% d'une durée de 1 seconde,
- et type C : creux de tension biphasé de 30% d'une durée de 1 seconde.

Comme le démontre la courbe de la figure 5.38, les cas de test choisis entraîneraient le délestage de l'équipement électrique car ils sont situés dans la partie inférieure de la courbe SEMI F47.

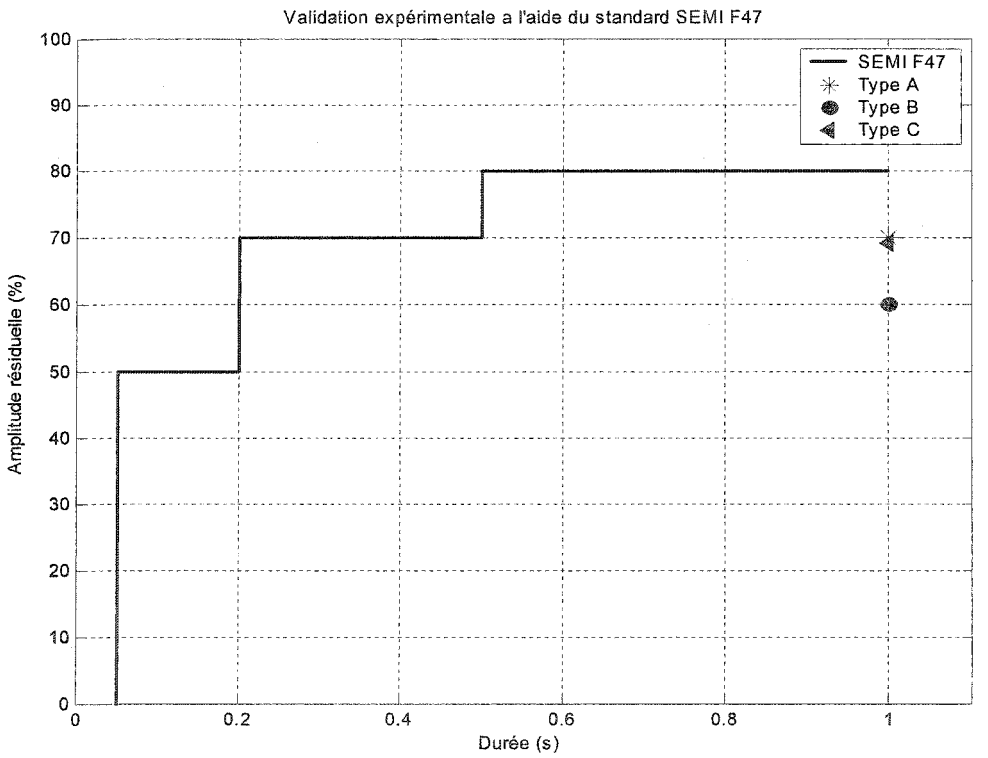


Figure 5.38 : SEMI F47 : cas de test pour l'expérimentation

La figure 5.39 présente les résultats obtenus en présence d'un creux de tension triphasé de 30% pendant 1 seconde (60 cycles).

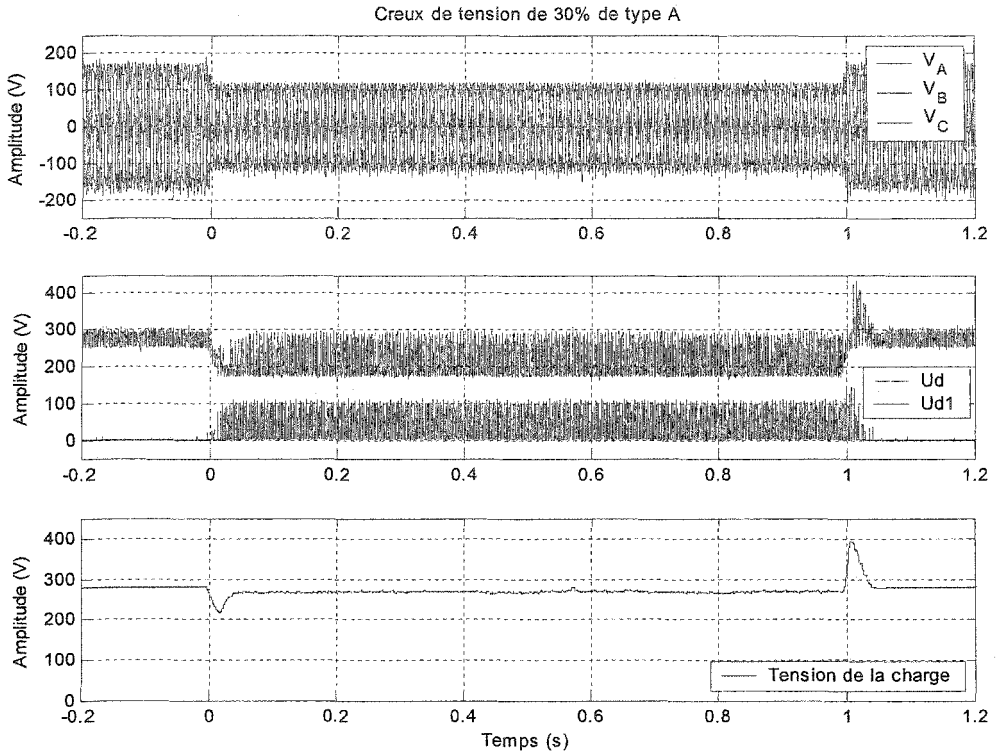


Figure 5.39 : Creux de tension de 30% de type A pendant 1 seconde

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

Afin de mieux appréhender les performances du système de compensation en régime transitoire, nous présentons aux figures 5.40 et 5.41 des agrandissements de la figure 5.39 au début et à la fin du creux de tension.

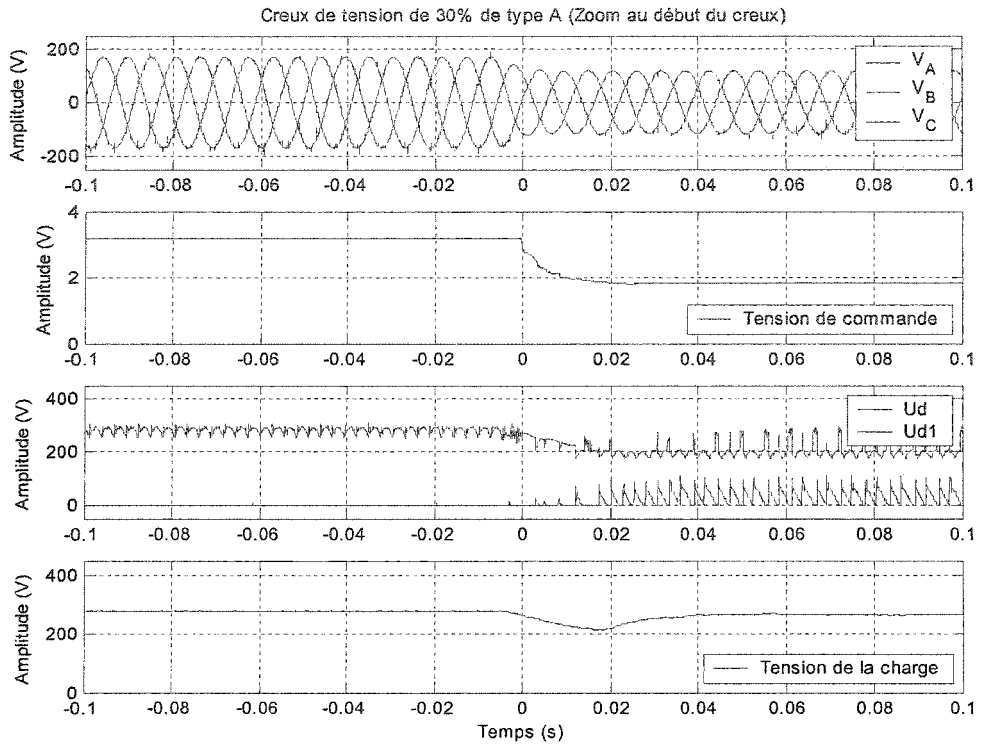


Figure 5.40 : Formes d'ondes de tensions mesurées au début d'un creux de tension de 30% de type A

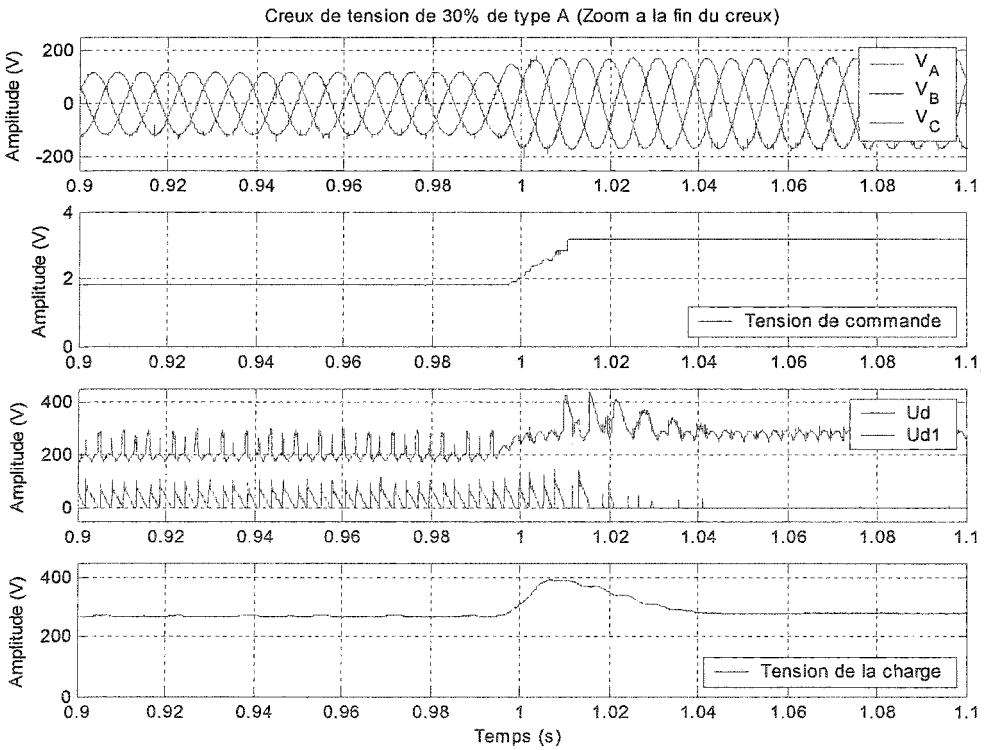


Figure 5.41 : Formes d'ondes de tensions mesurées à la fin d'un creux de tension de 30% de type A

Les résultats expérimentaux obtenus démontrent qu'il y a une compensation effective. Cependant, au début du creux, on note un délai de compensation équivalent à 0.7 cycle (11.5ms). Ce délai de compensation équivaut à l'écart temporel entre l'instant où le creux de tension est détecté et l'instant où la tension de commande atteint 90% de sa valeur cible.

La deuxième série de tests porte sur des creux de tension monophasés de 40% pendant 1 seconde (60 cycles). La figure 5.42 présente les résultats obtenus.

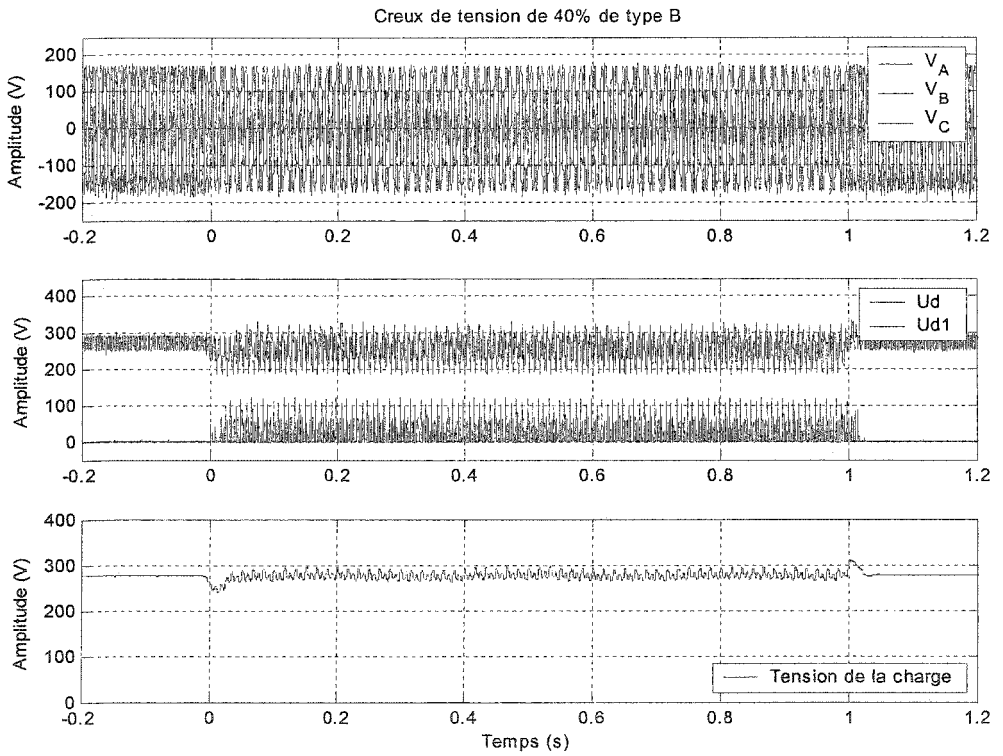


Figure 5.42 : Creux de tension de 40% de type B pendant 1 seconde

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

Les figures 5.43 et 5.44 présentent des agrandissements de la figure 5.42 au début et à la fin du creux de tension.

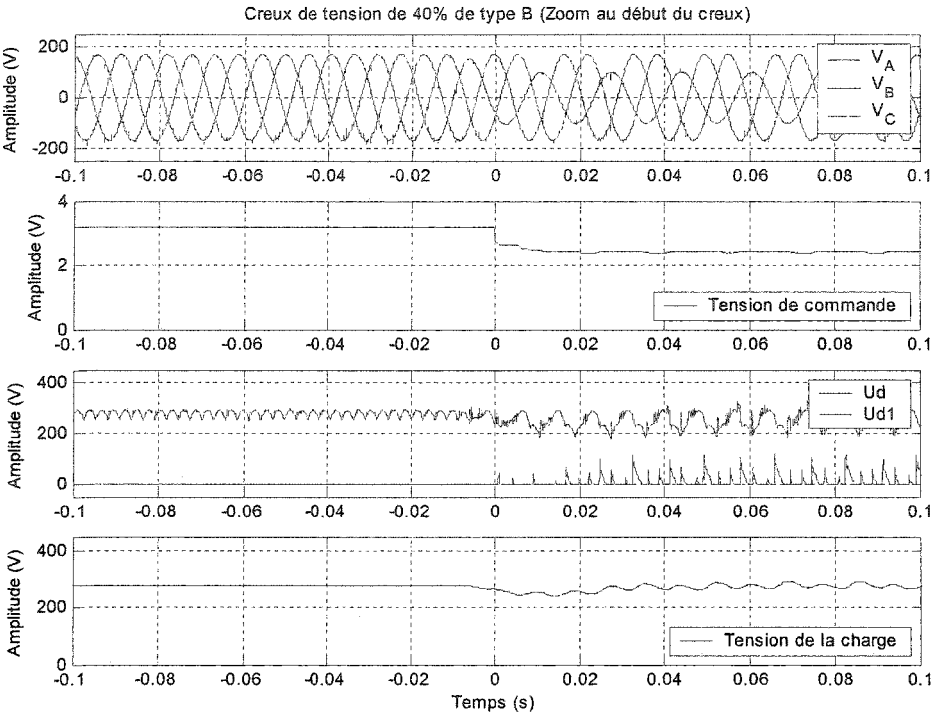


Figure 5.43 : Formes d'ondes de tensions mesurées au début d'un creux de tension de 40% de type B

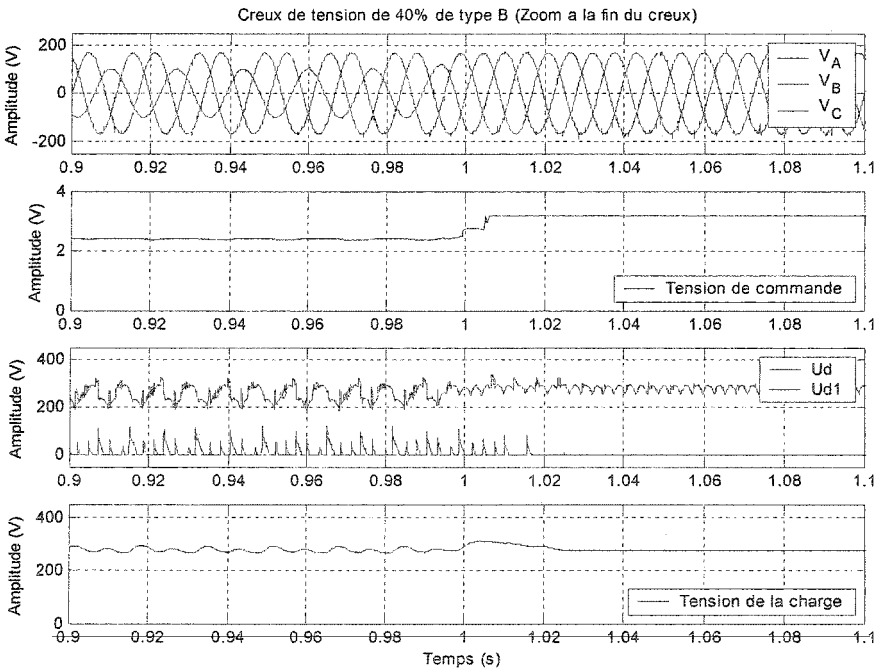


Figure 5.44 : Formes d'ondes de tensions mesurées à la fin d'un creux de tension de 40% de type B

Le délai de compensation au début du creux de tension est égal à 0.44 cycle, soit 7.4ms. La dernière série de tests porte sur des creux de tension biphasés de 30% pendant 1 seconde (60 cycles). La figure 5.45 présente les résultats obtenus.

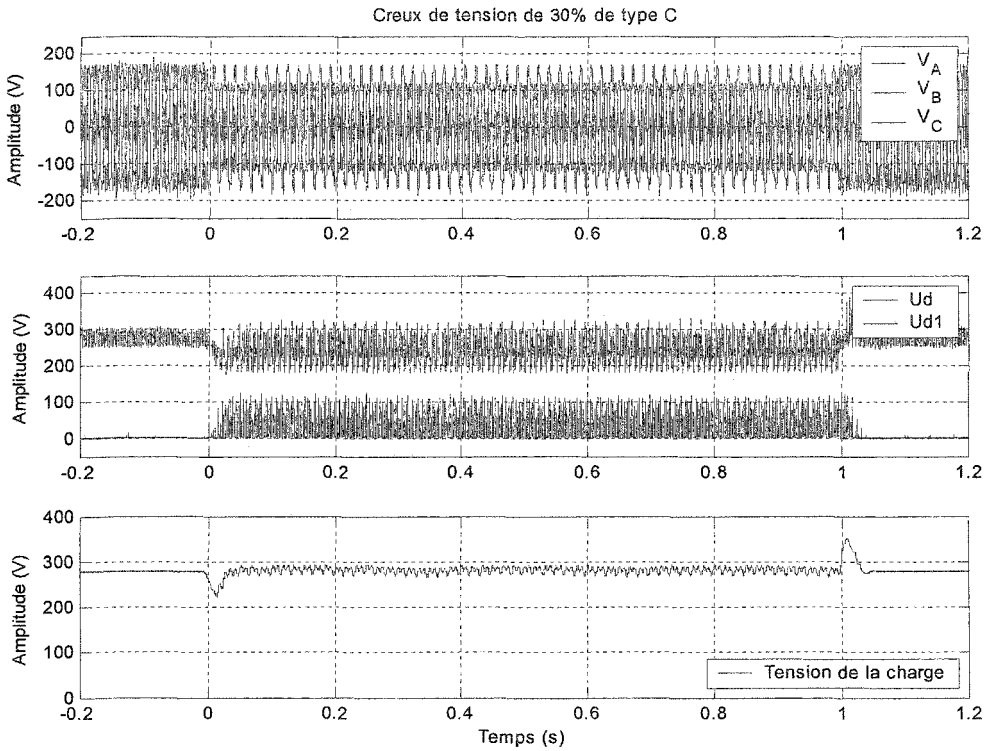


Figure 5.45 : Creux de tension de 30% de type C pendant 1 seconde

a) Tensions triphasées de la source
b) U_d, U_{d1} : Sortie principale, sortie du circuit auxiliaire de compensation
c) V_d : Tension aux bornes de la charge

Les figures 5.46 et 5.47 présentent des agrandissements de la figure 5.45 au début et à la fin du creux de tension.

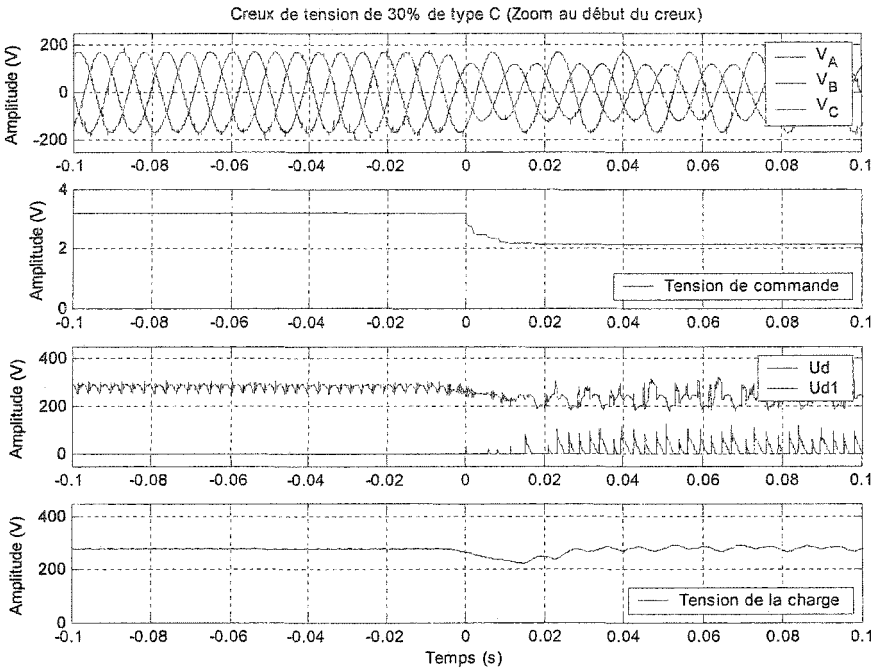


Figure 5.46 : Formes d'ondes de tensions mesurées au début d'un creux de tension de 30% de type C

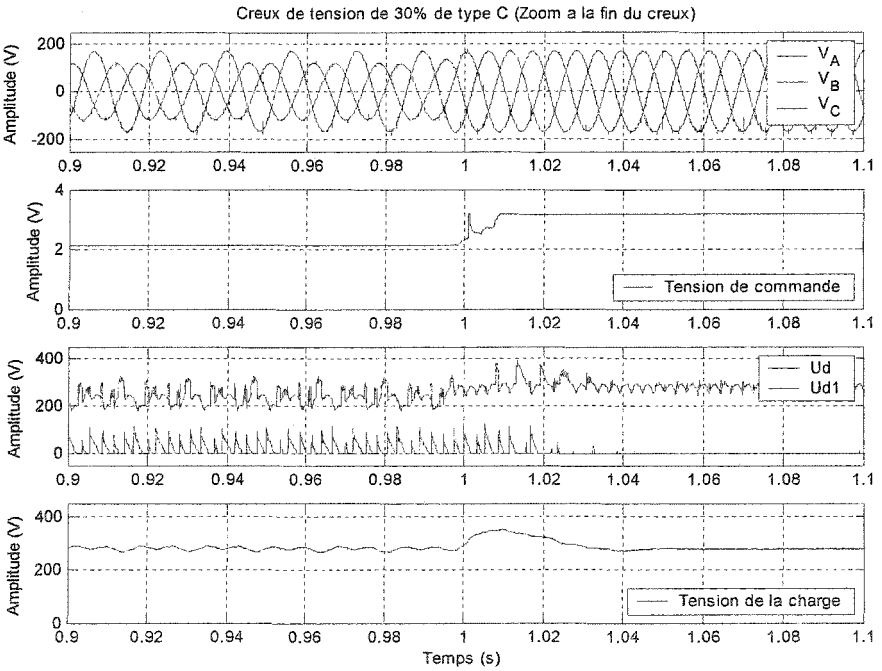


Figure 5.47 : Formes d'ondes de tensions mesurées à la fin d'un creux de tension de 30% de type C

Le délai de compensation au début du creux de tension est égal à 0.53 cycle, soit 8.88ms.

On note que la tension aux bornes de la charge a la même allure pour les creux de tension monophasés et biphasés. La raison est qu'à cause du transformateur Y-Δ, les creux de tension monophasés sont systématiquement transformés en creux de tension biphasés et les creux biphasés demeurent biphasés.

En considérant, les délais de compensation au regard du standard SEMI F47 (voir figure 5.48), on note que le système de compensation contribue largement à diminuer la sensibilité des équipements électriques avec redresseur triphasé à diodes face aux creux de tension équilibrés et déséquilibrés.

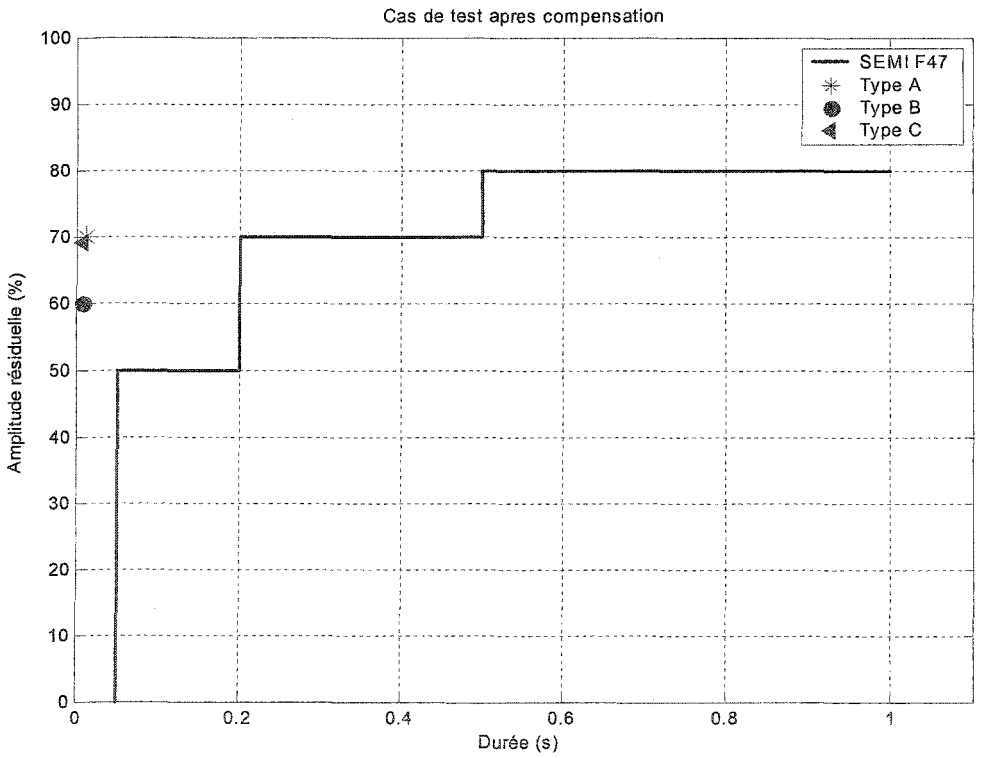


Figure 5.48 : SEMI F47 : performances du système auxiliaire de compensation

En dépit de l'handicap qu'est le temps de réponse naturelle des thyristors vis-à-vis de la tension de commande, les critères du standard SEMI F47 sont rencontrés car au

bout de 0.03 seconde, quelque soit le type de creux de tension, la tension du lien cc est ramenée à sa valeur nominale.

5.6 Conclusion

Nous avons présenté l'environnement expérimental ainsi que l'implantation des algorithmes de détection de creux de tension et de la loi de compensation. Nous avons présenté la conception des divers circuits électroniques de commande. Les résultats expérimentaux ont été également présentés.

La méthode d'injection du troisième harmonique nous a permis d'améliorer le THD en courant de l'ordre 48.5%. Cette méthode possède un avantage de taille : la grande simplicité de son circuit.

À l'aide d'une légère modification de topologie extrêmement simple à implanter et d'une loi de compensation basée sur la compensation de la valeur moyenne, les équipements électriques avec redresseur en pont de diodes ne sont plus sensibles aux effets de creux de tension équilibrés et déséquilibrés[3].

[3] Ils le sont dans la grande majorité.

CHAPITRE 6 : CONCLUSION ET RECOMMANDATIONS

Un système auxiliaire de compensation de creux de tension a été développé. Ce système auxiliaire s'adapte aux installations électriques avec redresseur en pont triphasé à diodes. Il est composé de deux sous-systèmes : (a) un redresseur à thyristors branché en série avec le redresseur à diodes permettant de désensibiliser les équipements électriques face aux creux de tension équilibrés et déséquilibrés; (b) un filtre quasi-passif qui permet d'améliorer le taux de distorsion global du courant dans la ligne grâce à la méthode d'injection du troisième harmonique. Un transformateur Y-Δ sert d'interface entre les deux sous-systèmes et le réseau.

6.1 Modification de la topologie du convertisseur

Afin de rendre les équipements électriques avec redresseur à diodes en pont triphasé plus robustes face aux déformations que peut subir la tension d'alimentation telles que les creux de tension, nous avons réalisé une modification de la topologie du convertisseur proposée par Xu et *al.*[27]. Cette modification procure à l'équipement électrique avec redresseur à diodes en pont triphasé une insensibilité face aux creux de tension équilibrés et déséquilibrés. Cette approche présente un certain nombre d'avantages :

(i) Faible coût dû à la simplicité du système auxiliaire proposé,

(ii) Pas des modifications majeures sur l'équipement standard,

(iii) Pas d'équipements de stockage tels que les batteries et les supercondensateurs,

(iv) N'affecte pas les autres charges connectées au PCC,

(v) Faible coût lié l'emplacement de la solution proposée (voir figure 6.1).

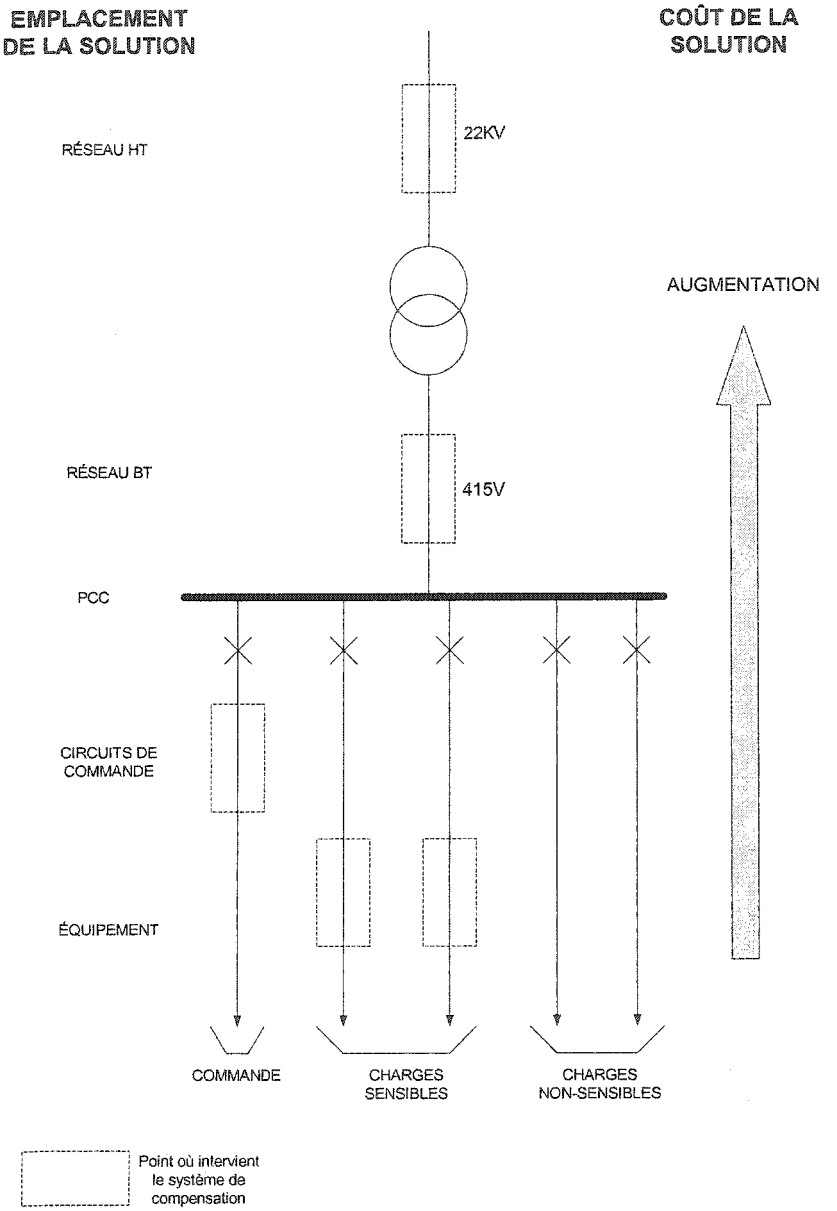


Figure 6.1 : Coût du système en fonction de l'emplacement de la solution

6.2 Détection et estimation des creux de tension

La théorie de la méthode Adaline appliquée à la détection et l'estimation des creux de tension a été exposée. À l'aide de la supervision de l'erreur entre le signal mesuré et le signal estimé, la vitesse de convergence de l'algorithme a été améliorée ; d'où une détection rapide de creux de tension. Cette supervision a permis également d'améliorer la précision pour l'estimation des creux de tension.
L'implantation de cet algorithme de détection de creux de tension a été faite dans un processeur numérique pour une évaluation en temps réel. Les performances en terme de rapidité de détection sont de 6.8ms en pratique alors que des simulations annoncaient un temps de 0.83ms.
Il faut souligner que le code implanté est généré automatiquement à partir des modèles Simulink (S-Functions); or ce code n'est pas optimisé [37]. Un exemple d'application présenté en [37], démontre une amélioration de 37.2% du temps d'exécution lors d'une implantation avec un code écrit manuellement. Une implantation de cet algorithme dans un DSP autonome et un meilleur pas de calcul, nous permettraient d'améliorer sans aucun doute les performances obtenues.

6.3 Loi de commande

La loi de commande implantée dans le cadre de ce travail est basée sur la compensation de la valeur moyenne de la tension.

6.4 Domaine d'application et développements futurs

Le système auxiliaire de compensation de creux de tension que nous avons réalisé trouve son application dans tous les équipements ou installations électriques possédant un redresseur en pont triphasé à diodes. Un exemple poignant est l'entraînement à vitesse variable dont nous avons évoqué la grande sensibilité face aux creux de tension.

Le prototype développé et l'ensemble des tests effectués nous ont permis de valider l'efficacité et la robutesse de notre solution.

Cependant ce protoype est loin d'être un produit fini; des améliorations doivent être apportées sur l'implantation des algorithmes de détection et d'estimation des creux de tension, l'implantation de la commande individuelle utilisée pour le pont à thyristors et sur la structure de la boucle de commande.

Pour l'implantation des algorithmes de détection, nous recommanderions une implantation sur un DSP autonome. Le code en langage C doit être directement implanté dans le DSP et non à l'aide de Matlab/Simulink.

La commande individuelle utilisée pour amorçer le pont à thyristors serait implantée dans un circuit FPGA. Cette option offre beaucoup plus de flexibité. Elle permettrait d'obtenir de meilleures performances dynamiques grâce à un parallélisme accru des opérations et une réduction significative de la dimension de la carte de commande et des coûts car au lieu des deux DSP comme c'est actuellement le cas, un seul circuit FPGA serait requis.

Enfin, une boucle de commande fermée est envisageable mais elle n'est pas aussi essentielle que les deux améliorations précedemment citées. Une boucle fermée de type proportionnel intégral permettrait de diminuer le délai de compensation en améliorant le temps de réponse de la commande et de limiter la surcompensation observée à la fin d'un creux de tension.

RÉFÉRENCES

[1] Santoso, S., Grady, W.M., Powers, E.J., Lamoree, J., Bhatt, S.C. *Characterization of distribution power quality events with Fourier and wavelet transforms.* IEEE Transactions on Power Delivery, Vol.15, No.1, 2000, pp. 247-254.

[2] Sannino, A., Miller, M.G., Bollen, M.H.J. *Overview of voltage sag mitigation.* IEEE Power Engineering Society Winter Meeting, Vol.4, 2000, pp.2872 – 2878.

[3] Bollen, M. H.J. *Understanding power quality problems: voltage sags and interruptions.* New-York : IEEE Press, 2000.

[4] Hamid, E.Y., Zen-Ichiro Kawasaki, Mardiana, R. *Wavelet packet transform for RMS and power measurements.* IEEE Power Engineering Society Summer Meeting, 2001, Vol.2, 2001, pp. 1243 - 1245

[5] Theerayuth Chatchanayuenyong, Chaipat Wattanasan. *Power quality improvement using a series active filter.* Université de Khonkaen - Département de génie électrique, 2000.

[6] Éloi Ngandui. *Contribution à la réduction des harmoniques non caractéristiques produits par les convertisseurs CA/CC.* Thèse, Ecole Polytechnique de Montréal, 1996.

[7] Alali, M.A.E., Saadate, S., Chapuis, Y.A., Braun, F. *Energetic study of a series active conditioner compensating voltage dips, unbalanced voltage and voltage harmonics.* VII IEEE International Power Electronics Congress, 2000, pp.80 –86

[8] Tunaboylu, N.S., Collins, E.R., Jr., Middlekauff, S.W., Morgan, R.L. *Ride-through issues for DC motor drives during voltage sags.* Proceedings of the IEEE, 1995, pp.52-58.

[9] Van Zyl, A., Spee, R., Faveluke, A., Bhowmik, S. *Voltage sag ride-through for adjustable-speed drives with active rectifiers.* IEEE Transactions on Industry Applications, 1998, Vol. 34, pp.1270–1277.

[10] Duran-Gomez, J.L., Enjeti, P.N., Byeong Ok Woo. *Effect of voltage sags on adjustable-speed drives: a critical evaluation and an approach to improve performance.* IEEE Transactions on Industry Applications, 1999, Vol. 35, No. 6, pp.1440 – 1449.

[11] K. Stockman, M. Didden, F. D'Hulster, R. Belmans. *Embedded solutions to protect textile processes against voltage sags,* IEEE Industry Applications Society, Octobre 2002.

[12] E. Ngandui, G. Olivier, Georges-Émile April. *Analyse des performances des redresseurs à thyristors en régime déséquilibré.* IEEE Canadian Conference on Electrical and Computer Engineering, 1999, pp.1234-1239.

[13] M. Sakui, H. Fujita, M. Shioya. *A method for calculating harmonic currents of a three-phase bridge uncontrolled rectifier with dc filter.* IEEE Transactions on Industrial Electronics, 1989, Vol.36, No.3, pp.434-440.

[14] V. E Wagner, A. A. Andreshak, and J. P. Staniak. *Power quality and factory automation.* IEEE Transactions on Industry Applications, 1990, Vol. 26, pp.620–626.

[15] E. R. Collins, Jr. and A. Mansoor. *Effects of voltage sags on ac motor drives.* IEEE Annual in Textile, Fiber, and Film Industry Technical Conference, 1997, pp.1-7.

[16] Middlekauff, S.W., Collins, E.R., Jr. *System and customer impact: considerations for series custom power devices.* IEEE Transactions on Power Delivery, 1998, Vol.13, No.1, pp.278 – 282.

[17] Tore M. Undeland, Ned Mohan, W.P. Robbins. *Power Electronics : Converters, Applications and Design 3e ed.* New-York : JohnWiley & Sons, Inc.

[18] Guy Seguier. *Les convertisseurs de l'électronique de puissance: La conversion alternatif-continu.* Technique et documentation-Lavoisier Paris, 1984.

[19] Dash, P.K., Swain, D.P., Liew, A.C., Rahman, S. *An adaptive linear combiner for on-line tracking of power system harmonics.* IEEE Transactions on Power Systems, 1996, Vol.11, No.4, pp.1730 – 1735.

[20] Abdel-Galil, T.K., El-Saadany, E.F., Salama, M.M.A. *Power Quality event detection using Adaline.* Electric Power Systems Research, 2002, pp.137-144.

[21] Pejovic, P., Janda, Z. *A novel harmonic-free three-phase diode bridge rectifier applying current injection.* Proceedings of the IEEE Applied Power Electronic Conference. 1999, pp. 241-247.

[22] Pejovic, P., Janda, Z. *A high power factor three-phase rectifier based on adaptative current injection applying buck converter.* 9th International Conference on Power Electronics and Motion Control – EPE-PEMC 2000. 2000, pp.140-144.

[23] Xu, J., Charette A., Rajagopalan V. *A quasi-passive modulation circuit improving the current waveform of conventional rectifiers.* 2000 IEEE 31st Annual Power Electronics Specialists Conference. 2000, pp.1463-1468.

[24] P. Pejovic. *Three-Phase high power factor rectifier based on the third harmonic current injection with passive resistance emulation.* 2000 IEEE 31st Annual Power Electronics Specialists Conference. 2000, pp.1463-1468.

[25] Janda, Z., Pejovic, P. *Multipulse high power factor rectifier applying a novel current injection network.* The 8th IEEE International Conference on Electronics, Circuits and Systems ICECS 2001. 2001, Vol.2, pp.651-654.

[26] Meza, J.C., Samra, A.H.. (1998). A new technique to reduce line-current harmonics generated by a three-phase bridge rectifier. Southeastcon '98. Proceedings, IEEE. pp 354-359.

[27] Xu, Y., K. Al-Haddad, P. Sicard, V. Rajagopalan, *A novel combined approach to voltage sag ride-through and current waveform improvement for adjustable-speed drives,* IEEE Power Electronics Specialist Conference PESC'2001, Vancouver CB, 17-21 juin 2001, Vol. 3, pp.1315-1320.

[28] Grafham, D. R., Golden, F. B. *SCR manual.* New York : General Electric Company, 1979.

[29] Malvino, A. P. *Principes d'électronique 3e édition.* Ediscience International, Paris, 1988.

[30] Newman, M.J., Holmes, D.G. *A universal custom power conditioner (UCPC) with selective harmonic voltage compensation.* IECON'02 Industrial Electronics Society. 2002, Vol.2, pp.1261 – 1266.

[31] Sarmiento, H.G., Estrada, E. *A voltage sag study in an industry with adjustable speed drives.* IEEE Industry Applications Magazine. Janvier/Février 1996, pp.16-19.

[32] El Shatshat, R., Kazerani, M., Salama, M.M.A. *Modular active power-line conditioner.* IEEE Transactions on Power Delivery, Vol.16, 2001, pp.700-709.

[33] El Shatshat, R., Kazerani, M., Salama, M.M.A. *Multi converter approach to active power filtering using current source converters.* IEEE Transactions on Power Delivery, Vol.16, 2001, pp.38-45.

[34] *PSIM Reference Manual,* Version 4.1, Powersim Technologies Inc., Vancouver, Canada, 1999.

[35] Kai Yao. *Modeling voltage sags on utility and industrial power systems.* Mémoire de maîtrise, Université de l'Alberta, Département de génie électrique et génie informatique, 2000.

[36] Texas Instruments. *TMS320LF/LC240xA DSP Controllers – Reference Guide – Systems and Peripherals. Application Report – SPRA357B.* TI, 2002.

[37] Ba-Razzouk A. *Introduction au système temps réel dSPACE.* Présentation - Séminaire de la Chaire sur la Puissance te l'Énergie Électrique. 21 Mars 2001.

ANNEXE A : DÉTECTION RAPIDE ET ESTIMATION DES CREUX DE TENSION PAR LA MÉTHODE ADALINE

Cette annexe présente le code Matlab de l'algorithme utilisé dans la détection et l'estimation des creux de tension. Cet algorithme utilise la méthode Adaline exposée au chapitre 3.

A-1 Algorithme de base

```
% ------------------------------------------------------------------
%  TITRE:                  RLS_algo1.m (sans supervision de l'erreur)
%  AUTEURS DE L'ALGORITHME :      Pierre Sicard  et Yi Zheng
%  DATE: Hiver 2002
%  DESCRIPTION:  Permet la détection des creux de tension
%                par la méthode Adaline. L'algorithme d'adaptation
%                des poids W est l'algorithme RLS
%  Copyright @ 2002 CPEE - LTE.
% ------------------------------------------------------------------

M=5000;        % Nombre d'itérations
N=5;           % Nombre d'harmoniques
lamda=0.96;    % facteur d'oubli
f=15360;       % fréquence d'échantillonnage f=15360Hz (256 ech.)

Ini_RLS=220;             % Valeur initiale de l'amplitude
noisemax=Ini_RLS*0.005  % L'amplitude max. du bruit est de 0.5% de l'amplitude du
signal
n=noisemax*randn(1,M);     % génère un bruit aléatoire
    %---------   ÉLABORATION DU SIGNAL MÉSURÉ -------------
pert=ones(1,M);
for (k=1:M)
    %expected output
    d1(k)=220*sin(2*pi*k*60/f+80/180*pi);            % fondamental
    if (k>M/20 & k<M/4)
        d1(k)=d1(k)*.7;
        pert(k)=.7;
    end
    if (k>M/3 & k<M/2)
        d1(k)=d1(k)*.5;
        pert(k)=.5;
    end
    if (k>M/1.5 & k<M/1.2)
        d1(k)=d1(k)*.25;
        pert(k)=.25;
    end
    d2(k)=220*0.05*sin(2*3*pi*k*60/f+60/180*pi);  %3e harmonique = 5% du fond.
```

```
    if (k>M/1.5 & k<M/1.3)                          %perturbation du 3e harm.
        d2(k)=d2(k)*.25;
    end
    d3(k)=220*0.025*sin(2*5*pi*k*60/f+(45/180)*pi); %5e harm. = 2.5% du fond.
    d4(k)=220*0.012*sin(2*7*pi*k*60/f+36/180*pi);   %7e harm. = 1.2% du fond.
    d5(k)=220*0.006*sin(2*9*pi*k*60/f+30/180*pi);   %9e harm. = 0.6% du fond.
    d_exp(k)=d1(k)+d2(k)+d3(k)+d4(k)+d5(k);         % signal mesuré
    d(k)=d_exp(k)+n(k);              % signal mesuré + bruit
end

        %---------  INITIALISATION -----------------

W(:,1)=zeros(2*N,1);                % initialisation de la matrice des poids W
K(:,1)=zeros(2*N,1);                % initialisation de la matrice des gains K
X(1,:)=zeros(1,2*N);                % initialisation du vecteur d'entrée X
P(:,:,1)=Ini_RLS*eye(2*N);          % initialisation de la matrice P
for j=2:N
    P(2*j-1,2*j-1,1)=Ini_RLS*0.05*(0.5^(j-2));
    P(2*j,2*j,1)=Ini_RLS*0.05*(0.5^(j-2));
end

for k=1:M;
    for t=1:N;
        Ins(2*t-1,k)= sin(2*(2*t-1)*pi*k*60/f);
        Ins(2*t,k)= cos(2*(2*t-1)*pi*k*60/f);
    end
end

        %-----------------  ALGORITHME RLS ----------
for(k=1:M)
    X(k,:)=Ins(:,k)';
    y(k)=X(k,:)*W(:,k); % évaluation de la sortie estimée y(k)
    e(k)=d(k)-y(k);     % calcul de l'erreur entre la sortie estimé et mesurée
    SE(k)=abs(e(k));
    K(:,k)=(P(:,:,k)*X(k,:)')/(lamda+X(k,:)*P(:,:,k)*X(k,:)');  % Calcul de la
matrice des gains d'adaptation K
    k=k+1;
    W(:,k)=W(:,k-1)+K(:,k-1)*e(k-1);  % Mise a jour de la matrice des poids W
% Calcul de la matrice d'autocorrelation P
    P(:,:,k)=(1/lamda)*(eye(2*N)-K(:,k-1)*X(k-1,:))*P(:,:,k-1);
end
pert(k)=pert(k-1);
for j=1:M;
    y1(j)=X(j,1)*W(1,j)+X(j,2)*W(2,j);  % évaluation du fondamental
    Amp(j)=sqrt(W(1,j)^2+W(2,j)^2);     % calcul de l'amplitude du fondamental
end
%---------------  AFFICHAGE  --------------------------

figure;
t=((1:M)/f)*60; % en cycle
plot(t,d1,t,y1(1:M));
title('Signal mesuré et signal estimé');
xlabel('temps en cycles');ylabel('Amplitude');
legend('fondamental mesuré','fondamental estimé');grid
figure;
plot(t,pert(1:M)*220,t,Amp(1:M));
title('Amplitude réelle et estimé');
xlabel('temps en cycles');ylabel('Amplitude');
legend('Amplitude réelle','Amplitude estimée');grid
figure;
semilogy(t,SE);grid
title('Erreur quadratique')
xlabel('temps en cycles');ylabel('Amplitude');
```

```
%--------- amplitudes et phases--------------
for j=1:N;
    ARLS(j)=sqrt(W(2*j-1,M)^2+W(2*j,M)^2);
    PRLS(j)=atan(W(2*j,M)/W(2*j-1,M))*180/pi;
end

P_exp=[80 60 45 36 30];                % amplitude et phase réelles
A_exp=[220 11 5.5 2.64 1.32];
if N>5,
    for i=6:N;
        P_exp(i)=0;
        A_exp(i)=0;
    end
end
phase=[P_exp' PRLS'];                  % amplitude et phase estimées
Amplitude=[A_exp' ARLS'];
disp(' ')
disp('amplitudes')
disp('Réelles        Estimées')
disp(Amplitude)
disp(' ')
disp('Phases')
disp('Réelles        Estimées')
disp(phase)
```

A-2 Algorithme modifié (avec supervision de l'erreur)

```
% ------------------------------------------------------------------
%  TITRE:                 RLS_algo2.m (avec supervision de l'erreur)
%  AUTEURS DE L'ALGORITHME :      Pierre Sicard  et Yi Zheng
%  DATE: Hiver 2002
%  DESCRIPTION:  Permet la détection des creux de tension
%                par la méthode Adaline. L'algorithme d'adaptation
%                des poids W est l'algorithme RLS
%  Copyright @ 2002 CPEE - LTE.
% ------------------------------------------------------------------


M=5000;         % Nombre d'itérations
N=5;            % Nombre d'harmoniques
lamda=0.99;     % facteur d'oubli
f=15360;        % fréquence d'échatillonnage f=15360Hz (256 ech.)
Ini_RLS=220;    % Valeur initiale de l'amplitude

THRESHOLD=1;
PERIOD=5;
eps=Ini_RLS*0.1;        % 10% de la valeur de l'amplitude nominale
noisemax=Ini_RLS*0.005;  % l'amplitude max. du bruit est de 0.5% de l'amplitude
du signal
n=noisemax*randn(1,M);                    % génère un bruit aléatoire
%--- Initialisation de matrice P
for j=2:N
    P(2*j-1,2*j-1,1)=Ini_RLS*0.05*(0.5^(j-2));
    P(2*j,2*j,1)=Ini_RLS*0.05*(0.5^(j-2));
end

Threshold=THRESHOLD;    % seuil d'erreur pour permettre la réinitialisation
period=PERIOD;          % nombre d'itération permise pour l'identification avant
une réinitialisation
setflag=0;              % si setflag=1, un creux est détecté sinon, il est a 0
```

```
%--------------------   ÉLABORATION DU SIGNAL MÉSURÉ ------------------
pert=ones(1,M);
for (k=1:M)
    d1(k)=220*sin(2*pi*k*60/f+80/180*pi);              % fondamental
    if (k>M/20 & k<M/4)
        d1(k)=d1(k)*.7;
        pert(k)=.7;
    end
    if (k>M/3 & k<M/2)
        d1(k)=d1(k)*.5;
        pert(k)=.5;
    end

    if (k>M/1.5 & k<M/1.2)
        d1(k)=d1(k)*.25;
        pert(k)=.25;
    end
    d2(k)=220*0.05*sin(2*3*pi*k*(60/f)+60/180*pi);  % 3e harm. = 5% du fond.
    if (k>M/1.5 & k<M/1.3)                           % perturbation du 3e harm.
        d2(k)=d2(k)*.25;
    end
    d3(k)=220*0.025*sin(2*5*pi*k*60/f+(45/180)*pi); %5e harm. = 2.5% du fond.
    d4(k)=220*0.012*sin(2*7*pi*k*60/f+36/180*pi);   %7e harm. = 1.2% du fond.
    d5(k)=220*0.006*sin(2*9*pi*k*60/f+30/180*pi);   %9e harm. = 0.6% du fond.
    d_exp(k)=d1(k)+d2(k)+d3(k)+d4(k)+d5(k);          % signal mesuré
    d(k)=d_exp(k)+n(k);          % signal mesuré + bruit
end

%-------------------- INITIALISATION ---------------------------------
W(:,1)=zeros(2*N,1);               % initialisation de la matrice des poids W
K(:,1)=zeros(2*N,1);               % initialisation de la matrice des gains K
X(1,:)=zeros(1,2*N);               % initialisation du vecteur d'entrée X
P(:,:,1)=Ini_RLS*eye(2*N);         % initialisation de la matrice P
% initialisation de la matrice des sin(wt), cos(wt), sin(3wt), cos(3wt)...
for k=1:M;
    for t=1:N;
        Ins(2*t-1,k)= sin(2*(2*t-1)*pi*k*60/f);
        Ins(2*t,k)= cos(2*(2*t-1)*pi*k*60/f);
    end
end
%-------------- ALGORITHME RLS  --------------------
for(k=1:M)
    X(k,:)=Ins(:,k)';
    y(k)=X(k,:)*W(:,k);  %évaluation de la sortie estimée y(k)
    e(k)=d(k)-y(k);      %calcul de l'erreur entre la sortie estimé et mesurée
    SE(k)=e(k)^2;
    %-------------- SUPERVISION APRES LE CALCUL DE L'ERREUR --------
    if ((abs(e(k))>eps)&(period==PERIOD))     % Si l'erreur est sup. à epsilon,
diminuer le seuil de réinitialisation de la matrice P
        Threshold=Threshold-1;
    else
        Threshold=THRESHOLD;
    end
    if(Threshold==0)                          % Si un creux de tension est détecté
        setflag=1;
    end
    %------- Réinitialisation de la matrice P ----------
    if (setflag==1)&(period==PERIOD)
        P(:,:,k)=Ini_RLS*eye(2*N);
        for j=2:N
            P(2*j-1,2*j-1,k)=Ini_RLS*0.05*(0.5^(j-2));
            P(2*j,2*j,k)=Ini_RLS*0.05*(0.5^(j-2));
        end
```

```
    end
    if(setflag==1)
        period=period-1;
        if(period==0)
            setflag=0;
            Threshold=THRESHOLD;
            period=PERIOD;
        end
    end
    %---------------- SUITE DE L'ALGORITHME RLS  --------------
    K(:,k)=(P(:,:,k)*X(k,:)')/(lamda+X(k,:)*P(:,:,k)*X(k,:)');  % Calcul de la
matrice des gains d'adaptation K
    k=k+1;
    W(:,k)=W(:,k-1)+K(:,k-1)*e(k-1);   % Mise a jour de la matrice des poids W
    % Calcul de la matrice d'autocorrelation P
    P(:,:,k)=(1/lamda)*(eye(2*N)-K(:,k-1)*X(k-1,:))*P(:,:,k-1);
end
pert(k)=pert(k-1);
for j=1:M;
    y1(j)=X(j,1)*W(1,j)+X(j,2)*W(2,j);  % évaluation du fondamental
    Amp(j)=sqrt(W(1,j)^2+W(2,j)^2);     % calcul de l'amplitude du fondamental
end

%---------------  AFFICHAGE  --------------------------
figure;
t=((1:M)/f)*60; % en cycle
plot(t,d1,t,y1(1:M));
title('Signal mesuré et signal estimé');
xlabel('temps en cycles');ylabel('Amplitude');
legend('fondamental mesuré','fondamental estimé');grid
figure;
plot(t,pert(1:M)*220,t,Amp(1:M));
title('Amplitude réelle et estimé');
xlabel('temps en cycles');ylabel('Amplitude');
legend('Amplitude réelle','Amplitude estimée');grid
figure;
semilogy(t,SE);grid
title('Erreur quadratique')
xlabel('temps en cycles');ylabel('Amplitude');

%--------- amplitudes et phases--------------
for j=1:N;
    ARLS(j)=sqrt(W(2*j-1,M)^2+W(2*j,M)^2);
    PRLS(j)=atan(W(2*j,M)/W(2*j-1,M))*180/pi;
end
P_exp=[80 60 45 36 30];              % amplitude et phase réelles
A_exp=[220 11 5.5 2.64 1.32];
if N>5,
    for i=6:N;
        P_exp(i)=0;
        A_exp(i)=0;
    end
end
phase=[P_exp' PRLS'];                % amplitude et phase estimées
Amplitude=[A_exp' ARLS'];
disp(' ')
disp('amplitudes')
disp('Réelles        Estimées')
disp(Amplitude)
disp(' ')
disp('Phases')
disp('Réelles        Estimées')
disp(phase)
```

ANNEXE B : IMPLANTATION SOUS PSIM DE L'ALGORITHME DE DÉTECTION DES CREUX DE TENSION

Cette annexe présente le code en langage C de l'algorithme utilisé dans la détection et l'estimation des creux de tension. Ce code permet à la compilation et à l'édition des liens, de générer une dll (Dynamic Link Library) utilisé lors de la simulation dans PSIM.

B-1 Estimateur d'amplitude de la phase A

```
/*-------------------------------------------------------
  Description: Estimateur d'amplitude de la phase A à
               l'aide de la méthode Adaline

  ========================================================
  HELP: COMMENT CREER UN DLL UTILISÉ SOUS PSIM
  ========================================================
  To compile the program into DLL:

  For Microsoft Visual C++ 5.0 or 6.0:
      - Create a directory called "C:\ms_user0", and copy the file "ms_user0.c"
        that comes with the PSIM software into the directory C:\ms_user0.
      - Start Visual C++. From the "File" menu, choose "New". In the "Projects"
page,
        select "Win32 Dynamic-Link Library", and set "Project name" as
        "ms_user0", and "Location" as "C:\ms_user0". Make sure that "Create
        new workspace" is selected, and under "Platform", "Win32" is selected.
      - [for Version 6.0] When asked "What kind of DLL would you like to
create?",
        select "An empty DLL project.".
      - From the "Project" menu, go to "Add to Project"/"Files...", and select
        "ms_user0.c".
      - From the "Build" menu, go to "Set Active Configurations...", and select
        "Win32 Release". From the "Build" menu, choose "Rebuild All" to generate
the
        DLL file "ms_user0.dll". The DLL file will be stored under the directory
        "C:\ms_user0\release".

  After the DLL file "ms_user0.dll" is generated, backup the default file
  into another file or directory, and copy your DLL file into the PSIM
  directory (and overwriting the existing file). You are then ready to run
  PSIM with your DLL.

  Activate (enable) the following line if the file is a C++ file
  (i.e. "ms_user0.cpp") extern "C"
```

```
 You may change the variable names (say from "t" to "Time").
  But DO NOT change the function name, number of variables, variable type,
  and sequence.

  Variables:
      t: Time, passed from PSIM by value
   delt: Time step, passed from PSIM by value
     in: input array, passed from PSIM by reference
    out: output array, sent back to PSIM (Note: the values of out[*] can
         be modified in PSIM)
 The maximum length of the input and output array "in" and "out" is 20.
 Warning: Global variables above the function ms_user0 (t,delt,in,out)
          are not allowed!!!
  ------------------------------------------------------*/

#include <math.h>

__declspec(dllexport) void ms_user0 (t, delt, in, out)

/*Note that all the variables must be defined as "double"*/

 double t, delt;
 double *in, *out;
{
        /* pour ce programme, N est 1, la frequence fondamentale est 60 Hz.*/

        #define N 1
        #define PERIOD 5
        #define THRESHOLD 1.
        const double lamda = 1, Ini_RLS = 200., eps = 0.1;
        const double f = 15360., pi = 3.14159265358979323846;
        static int ini_mode = 1, setflag = 0, period = PERIOD;
        static double Threshold = THRESHOLD;
        static double P[2*N][2*N], W[2*N];
        static double KP[2*N][2*N], P1[2*N][2*N],lamda1, y1, Amp;
        static double X[2*N], K[2*N], e, y;
        int i,j,k;

        /* Initialisation */
        if (ini_mode == 1)
        {
                ini_mode = 0;
                /* P=Ini_RLS*eye(2*N);*/
                for (i=0;i<2*N;i++)
                {
                        for (j=0;j<2*N;j++)
                        {
                                P[i][j] = 0.;
                                if (i==j) {P[i][j] = Ini_RLS;
                                }
                        }
                }
                for (j=2;j<=N;j++) {
                        P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                        P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
                }
        }
        /* Calculer X */
        for (i=1; i<=N; i++)
        {
                X[2*i-2]= sin(2*(2*(double)i-1)*pi*t*60.);
                X[2*i-1]= cos(2*(2*(double)i-1)*pi*t*60.);
        }
```

```
/* y=X*W;  */

y=0.;
for (i=0; i<2*N; i++)
{
        y = y + X[i]*W[i];
}
/* e=d(k)-y;       */

e = in[0] - y;
if ((fabs(e)>eps)&&(period==PERIOD))
{
        Threshold=Threshold -1.;
}
else
{
        Threshold=THRESHOLD;
}
if (Threshold==0.)
{
        setflag=1;
}

if ((setflag==1)&&(period==PERIOD))
{
        /* P=Ini_RLS*eye(2*N);*/
        for (i=0;i<2*N;i++)
        {
                for (j=0;j<2*N;j++)
                {
                        P[i][j] = 0.;
                        if (i==j)
                        {
                                P[i][j] = Ini_RLS;
                        }
                }
        }
        for (j=2;j<=N;j++)
        {
                P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
        }
}

if (setflag==1)
{
        period=period-1;
        if (period==0)
        {
                setflag=0;
                Threshold=THRESHOLD;
                period=PERIOD;
        }
}


/* RLS algorithm with forgetting factor */
/* K=(P*X')/(lamda+X*P*X'); */
for (i=0; i<2*N; i++)
{
        K[i]=0;
        for (j=0; j<2*N; j++)
        {
```

```
                K[i] = K[i] + P[i][j]*X[j];
            }
        }
        lamda1 = lamda;
        for (i=0; i<2*N; i++)
        {
            lamda1 = lamda1 + K[i]*X[i];
        }

        for (i=0; i<2*N; i++)
        {
            K[i] = K[i]/lamda1;
        }

          /* W=W+K*e;*/
        for (i=0; i<2*N; i++)
        {
            W[i] = W[i] + K[i]*e;
        }
        /* P=(1/lamda)*(eye(2*N)-K*X)*P;*/
        for (i=0; i<2*N; i++)
        {
            for (j=0; j<2*N; j++)
            {
                KP[i][j] = - K[i]*X[j];
                if (i==j)
                {
                    KP[i][j] = KP[i][j] + 1.;
                }
            }
        }
        for (i=0; i<2*N; i++)
        {
            for (j=0; j<2*N; j++)
            {
                P1[i][j] = 0.;
                for (k=0; k<2*N; k++)
                {
                    P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
                }
            }
        }
        for (i=0; i<2*N; i++)
        {
            for (j=0; j<2*N; j++)
            {
                P[i][j] = P1[i][j]/lamda;
            }
        }
        y1 = X[0]*W[0] + X[1]*W[1];
        Amp = sqrt(W[0]*W[0] + W[1]*W[1]);
        out[0] = y1;      /* Tension estimée */
        out[1] = Amp;     /* Amplitude estimée */
        out[2] = e;       /* erreur entre la tension mesurée et estimée */

}
```

B-2 Estimateur d'amplitude de la phase B

```
/*-----------------------------------------------------
  Description: Estimateur d'amplitude de la phase B à
                              l'aide de la méthode Adaline
  -----------------------------------------------------*/

#include <math.h>

__declspec(dllexport) void ms_user1 (t, delt, in, out)

/*Note that all the variables must be defined as "double"*/

 double t, delt;
 double *in, *out;
{
        /* pour ce programme, N est 1, la frequence fondamentale est 60 Hz.*/

        #define N 1
        #define PERIOD 5
        #define THRESHOLD 1.
        const double lamda = 1, Ini_RLS = 200., eps = 0.1;
        const double f = 15360., pi = 3.14159265358979323846;
        static int ini_mode = 1, setflag = 0, period = PERIOD;
        static double Threshold = THRESHOLD;

        static double P[2*N][2*N], W[2*N];
        static double KP[2*N][2*N], P1[2*N][2*N],lamda1, y1, Amp;
        static double X[2*N], K[2*N], e, y;
        int i,j,k;

        /* Initialisation */

        if (ini_mode == 1)
        {
                ini_mode = 0;
                /* P=Ini_RLS*eye(2*N);*/
                for (i=0;i<2*N;i++) {
                        for (j=0;j<2*N;j++) {
                                P[i][j] = 0.;
                                if (i==j) {P[i][j] = Ini_RLS;
                                }
                        }
                }
                for (j=2;j<=N;j++) {
                        P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                        P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
                }
        }

        /* Calculer X */
        for (i=1; i<=N; i++) {
                X[2*i-2]= sin(2*(2*(double)i-1)*pi*t*60.);
                X[2*i-1]= cos(2*(2*(double)i-1)*pi*t*60.);
        }
        /* y=X*W;  */
        y=0.;
        for (i=0; i<2*N; i++) {
                y = y + X[i]*W[i];
        }
```

```
/* e=d(k)-y;       */

e = in[0] - y;
if ((fabs(e)>eps)&&(period==PERIOD))
{
        Threshold=Threshold -1.;
}
else
{
        Threshold=THRESHOLD;
}
if (Threshold==0.) {
        setflag=1;
}

if ((setflag==1)&&(period==PERIOD))
{
        /* P=Ini_RLS*eye(2*N);*/
        for (i=0;i<2*N;i++)
        {
                for (j=0;j<2*N;j++)
                {
                        P[i][j] = 0.;
                        if (i==j)
                        {
                                P[i][j] = Ini_RLS;
                        }
                }
        }
        for (j=2;j<=N;j++)
        {
                P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
        }
}

if (setflag==1)
{
        period=period-1;
        if (period==0)
        {
                setflag=0;
                Threshold=THRESHOLD;
                period=PERIOD;
        }
}
/* RLS algorithm with forgetting factor */

   /* K=(P*X')/(lamda+X*P*X'); */
for (i=0; i<2*N; i++)
{
        K[i]=0;
        for (j=0; j<2*N; j++)
        {
                K[i] = K[i] + P[i][j]*X[j];
        }
}

lamda1 = lamda;
for (i=0; i<2*N; i++)
{
        lamda1 = lamda1 + K[i]*X[i];
```

```
        }

        for (i=0; i<2*N; i++)
        {
                K[i] = K[i]/lamda1;
        }

          /* W=W+K*e;*/

        for (i=0; i<2*N; i++)
        {
                W[i] = W[i] + K[i]*e;
        }

         /* P=(1/lamda)*(eye(2*N)-K*X)*P;*/

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        KP[i][j] = - K[i]*X[j];
                        if (i==j)
                        {
                                KP[i][j] = KP[i][j] + 1.;
                        }
                }
        }

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        P1[i][j] = 0.;
                        for (k=0; k<2*N; k++)
                        {
                                P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
                        }
                }
        }

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        P[i][j] = P1[i][j]/lamda;
                }
        }

        y1 = X[0]*W[0] + X[1]*W[1];
        Amp = sqrt(W[0]*W[0] + W[1]*W[1]);
        out[0] = y1;
        out[1] = Amp;
        out[2] = e;

}
```

B-3 Estimateur d'amplitude de la phase C

```
/*-----------------------------------------------------
  Description: Estimateur d'amplitude de la phase C à
                          l'aide de la méthode Adaline
  -----------------------------------------------------*/

#include <math.h>

__declspec(dllexport) void ms_user2 (t, delt, in, out)

/*Note that all the variables must be defined as "double"*/

 double t, delt;
 double *in, *out;
{
        /* pour ce programme, N est 1, la frequence fondamentale est 60 Hz.*/

        #define N 1
        #define PERIOD 5
        #define THRESHOLD 1.
        const double lamda = 1, Ini_RLS = 200., eps = 0.1;
        const double f = 15360., pi = 3.14159265358979323846;
        static int ini_mode = 1, setflag = 0, period = PERIOD;
        static double Threshold = THRESHOLD;

        static double P[2*N][2*N], W[2*N];
        static double KP[2*N][2*N], P1[2*N][2*N],lamda1, y1, Amp,Ref;
        static double X[2*N], K[2*N], e, y;
        int i,j,k;

        /* Initialisation */

        if (ini_mode == 1)
        {
                ini_mode = 0;
                /* P=Ini_RLS*eye(2*N);*/
                for (i=0;i<2*N;i++) {
                        for (j=0;j<2*N;j++) {
                                P[i][j] = 0.;
                                if (i==j) {P[i][j] = Ini_RLS;
                                }
                        }
                }
                for (j=2;j<=N;j++) {
                        P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                        P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
                }
        }

        /* Calculer X */
        for (i=1; i<=N; i++) {
                X[2*i-2]= sin(2*(2*(double)i-1)*pi*t*60.);
                X[2*i-1]= cos(2*(2*(double)i-1)*pi*t*60.);
        }
        /* y=X*W;  */

        y=0.;
        for (i=0; i<2*N; i++) {
                y = y + X[i]*W[i];
        }
```

```
/* e=d(k)-y;      */

e = in[0] - y;
if ((fabs(e)>eps)&&(period==PERIOD))
{
        Threshold=Threshold -1.;
}
else
{
        Threshold=THRESHOLD;
}
if (Threshold==0.) {
        setflag=1;
}

if ((setflag==1)&&(period==PERIOD))
{
        /* P=Ini_RLS*eye(2*N);*/
        for (i=0;i<2*N;i++)
        {
                for (j=0;j<2*N;j++)
                {
                        P[i][j] = 0.;
                        if (i==j)
                        {
                                P[i][j] = Ini_RLS;
                        }
                }
        }
        for (j=2;j<=N;j++)
        {
                P[2*j-2][2*j-2]=Ini_RLS*0.05*(pow(0.5,j-2));
                P[2*j-1][2*j-1]=Ini_RLS*0.05*(pow(0.5,j-2));
        }
}

if (setflag==1)
{
        period=period-1;
        if (period==0)
        {
                setflag=0;
                Threshold=THRESHOLD;
                period=PERIOD;
        }
}
/* RLS algorithm with forgetting factor */

   /* K=(P*X')/(lamda+X*P*X'); */
for (i=0; i<2*N; i++)
{
        K[i]=0;
        for (j=0; j<2*N; j++)
        {
                K[i] = K[i] + P[i][j]*X[j];
        }
}

lamda1 = lamda;
for (i=0; i<2*N; i++)
{
        lamda1 = lamda1 + K[i]*X[i];
}
```

```
        for (i=0; i<2*N; i++)
        {
                K[i] = K[i]/lamda1;
        }

          /* W=W+K*e;*/

        for (i=0; i<2*N; i++)
        {
                W[i] = W[i] + K[i]*e;
        }

         /* P=(1/lamda)*(eye(2*N)-K*X)*P;*/

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        KP[i][j] = - K[i]*X[j];
                        if (i==j)
                        {
                                KP[i][j] = KP[i][j] + 1.;
                        }
                }
        }

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        P1[i][j] = 0.;
                        for (k=0; k<2*N; k++)
                        {
                                P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
                        }
                }
        }

        for (i=0; i<2*N; i++)
        {
                for (j=0; j<2*N; j++)
                {
                        P[i][j] = P1[i][j]/lamda;
                }
        }

        y1 = X[0]*W[0] + X[1]*W[1];
        Amp = sqrt(W[0]*W[0] + W[1]*W[1]);
        out[0] = y1;
        out[1] = Amp;
        out[2] = e;

}
```

B-4 Estimateur de la valeur moyenne à la sortie d'un redresseur triphasé

```
/*-------------------------------------------------------
  Description : Permet d'estimer la valeur moyenne
                de la tension à la sortie d'un redresseur
                triphasé à diodes
 -------------------------------------------------------*/

#include <math.h>

__declspec(dllexport) void ms_user4 (t, delt, in, out)

/*Note that all the variables must be defined as "double" */

 double t, delt;
 double *in, *out;
{

        #define  pi       3.14159265358979323846

        const double eps = 0.000000000000000001;

        static double Van=1.0, Vbn=1.0, Vcn=1.0, Kan=1.0, Kbn=1.0, Kcn=1.0,
u=0.0, Vm=169.7056;
        static double t1=0, t3=0, t5=0, t7=0, t9=0, t11=0;
        static double A=0.0, B=0.0, C=0.0, Edo=280.681;
        static int sag=0;

        /* Initialisation */
        Van=in[0];
        Vbn=in[1];
        Vcn=in[2];

        Kan=Van/Vm;
        Kbn=Vbn/Vm;
        Kcn=Vcn/Vm;

        t1=atan2((sqrt(3)*Kcn),((2.0*Kan)+Kcn));
        t3=atan2((sqrt(3)*(Kcn+Kbn)),(Kcn-Kbn));
        t5=atan2((-sqrt(3)*Kbn),(2.0*Kan+Kbn)) + pi;
        t7=atan2((sqrt(3)*Kcn),(2.0*Kan+Kcn)) + pi;
        t9=atan2((sqrt(3)*(Kcn+Kbn)),(Kcn-Kbn)) + pi;
        t11=atan2((-sqrt(3)*Kbn),(2.0*Kan+Kbn)) + (2.0*pi);

        A=(1.0+cos(u))*(cos(t1)-cos(t5)-cos(t7)+cos(t11)) - sin(u)*(sin(t1)-
sin(t5)-sin(t7)+sin(t11));
        B=(cos(t3)+cos(t5)-cos(t9)-cos(t11))*(1+cos(u)-sqrt(3)*sin(u)) -
(sin(t3)+sin(t5)-sin(t9)-sin(t11))*(sqrt(3)*(1.0+cos(u))+sin(u));
        C=(cos(t1)+cos(t3)-cos(t7)-cos(t9))*(1+cos(u)+sqrt(3)*sin(u)) +
(sin(t1)+sin(t3)-sin(t7)-sin(t9))*(sqrt(3)*(1.0+cos(u))-sin(u));

        out[0]=(Vm/(8.0*pi))*((2.0*Kan*A) - Kbn*B + Kcn*C); /* Valeur moyenne */
        out[1]=280.681 - out[0];          /* Diminution de la veleur moyenne */

}
```

ANNEXE C : IMPLANTATION EN TEMPS-RÉEL DE L'ALGORITHME DE DÉTECTION DES CREUX DE TENSION

Cette annexe présente le code en langage C des différentes S-function utilisées lors de l'implantation dSPACE. Il s'agit des trois estimateurs d'amplitude et de l'estimateur de la valeur moyenne de la tension du lien cc.

C-1 Estimateur d'amplitude de la phase A

```
/*-------------------------------------------------------------------
 TITRE:              RLS_algo_a.c (Version dSPACE)
 AUTEURS DE L'ALGORITHME : Pierre Sicard
                           Yi Zheng
 DATE: Automne 2003
 DESCRIPTION: Permet la détection des creux de tension
                           par l'algorithme RLS

 S-FUNCTION écrit par: H. Joël Nanga Ndjana

 Copyright (c) 2003 CPEE - LTE.
-------------------------------------------------------------------*/


#define S_FUNCTION_NAME  RLS_algo_a
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include <math.h>

#define NUM_PARAMS (0)
#define M_PI        3.14159265358979323846

#define N 1
#define PERIOD 1
#define THRESHOLD 2

#define ENTREES 2
#define SORTIES 3


const double lamda_a = 0.97, Ini_RLS_a = 169.7056, eps_a = 5;
```

```
/*-----------------------------------------------------*/
  Permet de configurer les dimensions des vecteurs d'E/S
  -----------------------------------------------------*/

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S,  NUM_PARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
       {
             return;
       }

       ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;  /* Entrées */
    ssSetInputPortWidth(S, 0, ENTREES);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,1)) return;  /* Sorties */
    ssSetOutputPortWidth(S, 0, SORTIES);

    ssSetNumSampleTimes(S, 1);
}


/*-----------------------------------------------------*/
  Permet de spécifier que le pas d'échantillonnage utilisé
  sera celui spécifié dans l'outil de compilation
  -----------------------------------------------------*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs ================================================*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
      int i,j,k;
      static int ini_mode = 1, setflag = 0, period = PERIOD;
      static double Threshold = THRESHOLD;
      static double P[2*N][2*N]={169.7056,0.0,0.0,169.7056}, W[2*N];
      static double KP[2*N][2*N], P1[2*N][2*N],lamda1;
      static double X[2*N], K[2*N], e, y2;

   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T            *y    = ssGetOutputPortRealSignal(S,0);
   int_T             width = ssGetOutputPortWidth(S,0);

  /* Initialisation: */
           if (ini_mode == 1)
           {
                 ini_mode = 0;

                 for (i=0;i<2*N;i++)
                 {
                       for (j=0;j<2*N;j++)
                       {
                             P[i][j] = 0.;
                             if (i==j)
```

```
                        {
                                P[i][j] = Ini_RLS_a;
                        }
                }
        }

        for (j=2;j<=N;j++)
        {
                P[2*j-2][2*j-2]=Ini_RLS_a*0.05*(pow(0.5,j-2));
                P[2*j-1][2*j-1]=Ini_RLS_a*0.05*(pow(0.5,j-2));
        }
}

/* Calculer X */

        for (i=1; i<=N; i++)
        {
                X[2*i-2]= sin(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);
                X[2*i-1]= cos(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);

        }

        y2=0.0;

        for (i=0; i<2*N; i++)
        {
                y2 = y2 + X[i]*W[i];
        }
        e = *uPtrs[0] - y2;

        if ((fabs(e)> eps_a)&&(period==PERIOD))
        {
                Threshold=Threshold -1.0;
        }
        else
        {
                Threshold=THRESHOLD;
        }
        if (Threshold==0.0)
        {
                setflag=1;
        }

        if ((setflag==1)&&(period==PERIOD))
        {
                for (i=0;i<2*N;i++)
                {
                        for (j=0;j<2*N;j++)
                        {
                                P[i][j] = 0.0;
                                if (i==j)
                                {
                                        P[i][j] = Ini_RLS_a;
                                }
                        }
                }

                for (j=2;j<=N;j++)
                {
                        P[2*j-2][2*j-2]=Ini_RLS_a*0.05*(pow(0.5,j-2));
                        P[2*j-1][2*j-1]=Ini_RLS_a*0.05*(pow(0.5,j-2));
                }
        }
```

```
    if (setflag==1)
    {
          period=period-1;
          if (period==0)
              {
                    setflag=0;
                    Threshold=THRESHOLD;
                    period=PERIOD;
              }
    }

/* RLS algorithm with forgetting factor */

    for (i=0; i<2*N; i++)
    {
          K[i]=0;
          for (j=0; j<2*N; j++)
          {
                K[i] = K[i] + P[i][j]*X[j];
          }
    }
    lamda1 = lamda_a;
    for (i=0; i<2*N; i++)
    {
          lamda1 = lamda1 + K[i]*X[i];
    }
    for (i=0; i<2*N; i++)
    {
          K[i] = K[i]/lamda1;
    }
    for (i=0; i<2*N; i++)
    {
          W[i] = W[i] + K[i]*e;
    }

    for (i=0; i<2*N; i++)
    {
          for (j=0; j<2*N; j++)
          {
                KP[i][j] = - K[i]*X[j];
                if (i==j)
                {
                      KP[i][j] = KP[i][j] + 1.0;
                }
          }
    }

    for (i=0; i<2*N; i++)
    {
          for (j=0; j<2*N; j++)
          {
                P1[i][j] = 0.;
                for (k=0; k<2*N; k++)
                {
                      P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
                }
          }
    }

    for (i=0; i<2*N; i++)
    {
          for (j=0; j<2*N; j++)
```

```
                {
                        P[i][j] = P1[i][j]/lamda_a;
                }
        }

        y[0] = X[0]*W[0] + X[1]*W[1];
        y[1] = sqrt(W[0]*W[0] + W[1]*W[1]);
        y[2] = e;
}

static void mdlTerminate(SimStruct *S)
{
}


#if defined(MATLAB_MEX_FILE)
static void mdlSetInputPortComplexSignal(SimStruct *S, int_T port, int_T
iPortComplexSignal)
{
    int_T oPortComplexSignal = ssGetOutputPortComplexSignal(S,0);

    /* Set the complex signal of the input ports */
    ssSetInputPortComplexSignal(S, port, iPortComplexSignal);

    if(iPortComplexSignal == COMPLEX_YES)
       {
            /* Output port must be a complex signal */
        if(oPortComplexSignal == COMPLEX_INHERITED)
            {
           ssSetOutputPortComplexSignal(S, 0, COMPLEX_YES);
        }
            else if(oPortComplexSignal == COMPLEX_NO)
            {
                ssSetErrorStatus(S, "Output port must be complex.");
        }
    }
       else if(oPortComplexSignal != COMPLEX_NO)
       {
       /*
        * The current input port is a real signal.  If the other input port
        * is a real signal, the output port must be a real signal.
        */
       int_T otherPort = (port == 0)? 1 : 0;
       int_T otherPortComplexSignal = ssGetInputPortComplexSignal(S, otherPort);
       if(otherPortComplexSignal == COMPLEX_NO)
       {
            /* Both input ports are real signals */
            if(oPortComplexSignal == COMPLEX_INHERITED)
            {
                ssSetOutputPortComplexSignal(S, 0, COMPLEX_NO);
            }
            else if(oPortComplexSignal == COMPLEX_YES)
            {
                ssSetErrorStatus(S, "Output port must be real.");
          }
       }
    }
}

# define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL
static void mdlSetOutputPortComplexSignal(SimStruct *S, int_T port,
                                    int_T  oPortComplexSignal)
{
```

```
    /* Set the complex signal of the output ports */
    ssSetOutputPortComplexSignal(S, 0, oPortComplexSignal);

    if(oPortComplexSignal == COMPLEX_NO)
       {
        /* All inputs must be real */
        int_T i;

        for (i = 0; i < ENTREES; i++)
             {
            int_T iPortComplexSignal = ssGetInputPortComplexSignal(S, i);
            if(iPortComplexSignal == COMPLEX_INHERITED)
                    {
                ssSetInputPortComplexSignal(S, i, COMPLEX_NO);
            }
                    else if(iPortComplexSignal == COMPLEX_YES)
                    {
                           ssSetErrorStatus(S, "The output port is a 'real'
signal. "
                                  "All input ports must be 'real' signals.");
            }
        }
    }
       else
       {
             /* Output port is a complex signal.  Report an error, if all
         * inputs are real. */
             int_T i;
         boolean_T realInputs = true;
         for (i = 0; i < SORTIES; i++)
             {
            if(ssGetInputPortComplexSignal(S, i) != COMPLEX_NO)
                    {
                           realInputs = false;
                           break;
                    }
             }
             if(realInputs)
             {
            ssSetErrorStatus(S, "Input port and output port complex signal "
                           "mismatch. All input ports are 'real' signal. "
                           "The output port must be a 'real' signal.");
        }
    }
}
#endif /* MATLAB_MEX_FILE */

/* Required S-function trailer */

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

C-2 Estimateur d'amplitude de la phase B

```
/*-----------------------------------------------------------------------
 TITRE:             RLS_algo_b.c (Version dSPACE)
 AUTEURS DE L'ALGORITHME :Pierre Sicard, Yi Zheng
 DATE: Automne 2003
 DESCRIPTION:Permet la détection des creux de tension
                          par l'algorithme RLS

 S-FUNCTION écrit par: H. Joël Nanga Ndjana

 Copyright (c) 2003 CPEE - LTE.
-----------------------------------------------------------------------*/


#define S_FUNCTION_NAME  RLS_algo_b
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include <math.h>

#define NUM_PARAMS (0)
#define M_PI        3.14159265358979323846

#define N 1
#define PERIOD 1
#define THRESHOLD 2

#define ENTREES 2
#define SORTIES 3


const double lamda_b = 0.97, Ini_RLS_b = 169.7056, eps_b = 5;


/*-------------------------------------------------------/
  Permet de configurer les dimensions des vecteurs d'E/S
  -------------------------------------------------------*/

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S,  NUM_PARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
            return;
      }

      ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
       return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;  /* Entrées */
    ssSetInputPortWidth(S, 0, ENTREES);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,1)) return;  /* Sorties */
    ssSetOutputPortWidth(S, 0, SORTIES);
```

```
    ssSetNumSampleTimes(S, 1);
}


/*-----------------------------------------------------/
  Permet de spécifier que le pas d'échantillonnage utilisé
  sera celui spécifié dans l'outil de compilation
  -----------------------------------------------------*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs ============================================*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
      int i,j,k;
      static int ini_mode = 1, setflag = 0, period = PERIOD;
      static double Threshold = THRESHOLD;
      static double P[2*N][2*N]={169.7056,0.0,0.0,169.7056}, W[2*N];
      static double KP[2*N][2*N], P1[2*N][2*N],lamda1;
      static double X[2*N], K[2*N], e, y2;

    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    int_T             width = ssGetOutputPortWidth(S,0);

   /* Initialisation: */
            if (ini_mode == 1)
            {
                  ini_mode = 0;

                  for (i=0;i<2*N;i++)
                  {
                        for (j=0;j<2*N;j++)
                        {
                              P[i][j] = 0.;
                              if (i==j)
                              {
                                    P[i][j] = Ini_RLS_b;
                              }
                        }
                  }

                  for (j=2;j<=N;j++)
                  {
                        P[2*j-2][2*j-2]=Ini_RLS_b*0.05*(pow(0.5,j-2));
                        P[2*j-1][2*j-1]=Ini_RLS_b*0.05*(pow(0.5,j-2));
                  }
            }


      /* Calculer X */

            for (i=1; i<=N; i++)
            {
                  X[2*i-2]= sin(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);
                  X[2*i-1]= cos(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);

            }
```

```
        y2=0.0;

        for (i=0; i<2*N; i++)
        {
              y2 = y2 + X[i]*W[i];
        }
        e = *uPtrs[0] - y2;

        if ((fabs(e)> eps_b)&&(period==PERIOD))
        {
              Threshold=Threshold -1.0;
        }
        else
        {
              Threshold=THRESHOLD;
        }
        if (Threshold==0.0)
        {
              setflag=1;
        }

        if ((setflag==1)&&(period==PERIOD))
        {
              for (i=0;i<2*N;i++)
              {
                    for (j=0;j<2*N;j++)
                    {
                          P[i][j] = 0.0;
                          if (i==j)
                          {
                                P[i][j] = Ini_RLS_b;
                          }
                    }
              }
              for (j=2;j<=N;j++)
              {
                    P[2*j-2][2*j-2]=Ini_RLS_b*0.05*(pow(0.5,j-2));
                    P[2*j-1][2*j-1]=Ini_RLS_b*0.05*(pow(0.5,j-2));
              }
        }

    if (setflag==1)
    {
          period=period-1;
          if (period==0)
              {
                    setflag=0;
                    Threshold=THRESHOLD;
                    period=PERIOD;
              }
    }

/* RLS algorithm with forgetting factor */

    for (i=0; i<2*N; i++)
    {
          K[i]=0;
          for (j=0; j<2*N; j++)
          {
                K[i] = K[i] + P[i][j]*X[j];
          }
    }
    lamda1 = lamda_b;
```

```
            for (i=0; i<2*N; i++)
            {
                  lamda1 = lamda1 + K[i]*X[i];
            }
            for (i=0; i<2*N; i++)
            {
                  K[i] = K[i]/lamda1;
            }
            for (i=0; i<2*N; i++)
            {
                  W[i] = W[i] + K[i]*e;
            }
            for (i=0; i<2*N; i++)
            {
                  for (j=0; j<2*N; j++)
                  {
                        KP[i][j] = - K[i]*X[j];
                        if (i==j)
                        {
                              KP[i][j] = KP[i][j] + 1.0;
                        }
                  }
            }

            for (i=0; i<2*N; i++)
            {
                  for (j=0; j<2*N; j++)
                  {
                        P1[i][j] = 0.;
                        for (k=0; k<2*N; k++)
                        {
                              P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
                        }
                  }
            }

            for (i=0; i<2*N; i++)
            {
                  for (j=0; j<2*N; j++)
                  {
                        P[i][j] = P1[i][j]/lamda_b;
                  }
            }
            y[0] = X[0]*W[0] + X[1]*W[1];
            y[1] = sqrt(W[0]*W[0] + W[1]*W[1]);
            y[2] = e;
}

static void mdlTerminate(SimStruct *S)
{
}


#if defined(MATLAB_MEX_FILE)
static void mdlSetInputPortComplexSignal(SimStruct *S, int_T port, int_T
iPortComplexSignal)
{
    int_T oPortComplexSignal = ssGetOutputPortComplexSignal(S,0);

    /* Set the complex signal of the input ports */
    ssSetInputPortComplexSignal(S, port, iPortComplexSignal);

    if(iPortComplexSignal == COMPLEX_YES)
```

```
        {
              /* Output port must be a complex signal */
          if(oPortComplexSignal == COMPLEX_INHERITED)
              {
             ssSetOutputPortComplexSignal(S, 0, COMPLEX_YES);
          }
              else if(oPortComplexSignal == COMPLEX_NO)
              {
                    ssSetErrorStatus(S, "Output port must be complex.");
          }
      }
        else if(oPortComplexSignal != COMPLEX_NO)
        {
        /*
         * The current input port is a real signal.  If the other input port
         * is a real signal, the output port must be a real signal.
         */
        int_T otherPort = (port == 0)? 1 : 0;
        int_T otherPortComplexSignal = ssGetInputPortComplexSignal(S, otherPort);
        if(otherPortComplexSignal == COMPLEX_NO)
        {
              /* Both input ports are real signals */
              if(oPortComplexSignal == COMPLEX_INHERITED)
              {
                    ssSetOutputPortComplexSignal(S, 0, COMPLEX_NO);
              }
              else if(oPortComplexSignal == COMPLEX_YES)
              {
                    ssSetErrorStatus(S, "Output port must be real.");
             }
        }
      }
}

# define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL
static void mdlSetOutputPortComplexSignal(SimStruct *S, int_T port,
                                          int_T  oPortComplexSignal)
{
    /* Set the complex signal of the output ports */
    ssSetOutputPortComplexSignal(S, 0, oPortComplexSignal);

    if(oPortComplexSignal == COMPLEX_NO)
       {
        /* All inputs must be real */
        int_T i;

        for (i = 0; i < ENTREES; i++)
             {
            int_T iPortComplexSignal = ssGetInputPortComplexSignal(S, i);
            if(iPortComplexSignal == COMPLEX_INHERITED)
                  {
                ssSetInputPortComplexSignal(S, i, COMPLEX_NO);
            }
                  else if(iPortComplexSignal == COMPLEX_YES)
                  {
                        ssSetErrorStatus(S, "The output port is a 'real'
signal. "
                                    "All input ports must be 'real' signals.");
            }
        }
    }
       else
       {
```

```
            /* Output port is a complex signal.  Report an error, if all
         * inputs are real. */
            int_T i;
        boolean_T realInputs = true;
        for (i = 0; i < SORTIES; i++)
            {
            if(ssGetInputPortComplexSignal(S, i) != COMPLEX_NO)
                    {
                            realInputs = false;
                            break;
                    }
            }
            if(realInputs)
            {
            ssSetErrorStatus(S, "Input port and output port complex signal "
                            "mismatch. All input ports are 'real' signal. "
                            "The output port must be a 'real' signal.");
        }
    }
}
#endif /* MATLAB_MEX_FILE */

/* Required S-function trailer */

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

C-3 Estimateur d'amplitude de la phase C

```
/*---------------------------------------------------------------------
 TITRE:              RLS_algo_c.c (Version dSPACE)
 AUTEURS DE L'ALGORITHME :Pierre Sicard
                                                Yi Zheng
 DATE: Automne 2003
 DESCRIPTION:Permet la détection des creux de tension
                            par l'algorithme RLS

 S-FUNCTION écrit par: H. Joël Nanga Ndjana

 Copyright (c) 2003 CPEE - LTE.
---------------------------------------------------------------------*/


#define S_FUNCTION_NAME  RLS_algo_c
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include <math.h>

#define NUM_PARAMS (0)
#define M_PI        3.14159265358979323846

#define N 1
#define PERIOD 1
#define THRESHOLD 2
```

```
#define ENTREES 2
#define SORTIES 3

const double lamda_c = 0.97, Ini_RLS_c = 169.7056, eps_c = 5;

/*-----------------------------------------------------/
  Permet de configurer les dimensions des vecteurs d'E/S
  -----------------------------------------------------*/

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S,  NUM_PARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
            return;
      }

      ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;  /* Entrées */
    ssSetInputPortWidth(S, 0, ENTREES);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,1)) return;  /* Sorties */
    ssSetOutputPortWidth(S, 0, SORTIES);

    ssSetNumSampleTimes(S, 1);
}


/*-----------------------------------------------------/
  Permet de spécifier que le pas d'échantillonnage utilisé
  sera celui spécifié dans l'outil de compilation
  -----------------------------------------------------*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


/* Function: mdlOutputs ================================================*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
      int i,j,k;
      static int ini_mode = 1, setflag = 0, period = PERIOD;
      static double Threshold = THRESHOLD;
      static double P[2*N][2*N]={169.7056,0.0,0.0,169.7056}, W[2*N];
      static double KP[2*N][2*N], P1[2*N][2*N],lamda1;
      static double X[2*N], K[2*N], e, y2;

    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    int_T             width = ssGetOutputPortWidth(S,0);

   /* Initialisation: */
            if (ini_mode == 1)
            {
                  ini_mode = 0;
```

```
            for (i=0;i<2*N;i++)
            {
                  for (j=0;j<2*N;j++)
                  {
                        P[i][j] = 0.;
                        if (i==j)
                        {
                              P[i][j] = Ini_RLS_c;
                        }
                  }
            }

            for (j=2;j<=N;j++)
            {
                  P[2*j-2][2*j-2]=Ini_RLS_c*0.05*(pow(0.5,j-2));
                  P[2*j-1][2*j-1]=Ini_RLS_c*0.05*(pow(0.5,j-2));
            }
      }


/* Calculer X */

      for (i=1; i<=N; i++)
      {
            X[2*i-2]= sin(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);
            X[2*i-1]= cos(2*(2*(double)i-1)*M_PI*(*uPtrs[1])*60.0);

      }

      y2=0.0;

      for (i=0; i<2*N; i++)
      {
            y2 = y2 + X[i]*W[i];
      }
      e = *uPtrs[0] - y2;

      if ((fabs(e)> eps_c)&&(period==PERIOD))
      {
            Threshold=Threshold -1.0;
      }
      else
      {
            Threshold=THRESHOLD;
      }
      if (Threshold==0.0)
      {
            setflag=1;
      }

      if ((setflag==1)&&(period==PERIOD))
      {
            for (i=0;i<2*N;i++)
            {
                  for (j=0;j<2*N;j++)
                  {
                        P[i][j] = 0.0;
                        if (i==j)
                        {
                              P[i][j] = Ini_RLS_c;
                        }
                  }
            }
```

```
            for (j=2;j<=N;j++)
            {
                  P[2*j-2][2*j-2]=Ini_RLS_c*0.05*(pow(0.5,j-2));
                  P[2*j-1][2*j-1]=Ini_RLS_c*0.05*(pow(0.5,j-2));
            }
      }

   if (setflag==1)
   {
         period=period-1;
         if (period==0)
            {
                  setflag=0;
                  Threshold=THRESHOLD;
                  period=PERIOD;
            }
   }

/* RLS algorithm with forgetting factor */

   for (i=0; i<2*N; i++)
   {
         K[i]=0;
         for (j=0; j<2*N; j++)
         {
               K[i] = K[i] + P[i][j]*X[j];
         }
   }
   lamda1 = lamda_c;
   for (i=0; i<2*N; i++)
   {
         lamda1 = lamda1 + K[i]*X[i];
   }
   for (i=0; i<2*N; i++)
   {
         K[i] = K[i]/lamda1;
   }
   for (i=0; i<2*N; i++)
   {
         W[i] = W[i] + K[i]*e;
   }

   for (i=0; i<2*N; i++)
   {
         for (j=0; j<2*N; j++)
         {
               KP[i][j] = - K[i]*X[j];
               if (i==j)
               {
                     KP[i][j] = KP[i][j] + 1.0;
               }
         }
   }
   for (i=0; i<2*N; i++)
   {
         for (j=0; j<2*N; j++)
         {
               P1[i][j] = 0.;
               for (k=0; k<2*N; k++)
               {
                     P1[i][j] = P1[i][j] + KP[i][k]*P[k][j];
               }
         }
```

```
        }
        for (i=0; i<2*N; i++)
        {
              for (j=0; j<2*N; j++)
              {
                    P[i][j] = P1[i][j]/lamda_c;
              }
        }
        y[0] = X[0]*W[0] + X[1]*W[1];
        y[1] = sqrt(W[0]*W[0] + W[1]*W[1]);
        y[2] = e;
}


static void mdlTerminate(SimStruct *S)
{
}

#if defined(MATLAB_MEX_FILE)
static void mdlSetInputPortComplexSignal(SimStruct *S, int_T port, int_T
iPortComplexSignal)
{
    int_T oPortComplexSignal = ssGetOutputPortComplexSignal(S,0);

    /* Set the complex signal of the input ports */
    ssSetInputPortComplexSignal(S, port, iPortComplexSignal);

    if(iPortComplexSignal == COMPLEX_YES)
       {
             /* Output port must be a complex signal */
        if(oPortComplexSignal == COMPLEX_INHERITED)
             {
            ssSetOutputPortComplexSignal(S, 0, COMPLEX_YES);
        }
             else if(oPortComplexSignal == COMPLEX_NO)
             {
                   ssSetErrorStatus(S, "Output port must be complex.");
        }
    }
       else if(oPortComplexSignal != COMPLEX_NO)
       {
       /*
        * The current input port is a real signal.  If the other input port
        * is a real signal, the output port must be a real signal.
        */
       int_T otherPort = (port == 0)? 1 : 0;
       int_T otherPortComplexSignal = ssGetInputPortComplexSignal(S, otherPort);
       if(otherPortComplexSignal == COMPLEX_NO)
       {
             /* Both input ports are real signals */
             if(oPortComplexSignal == COMPLEX_INHERITED)
             {
                   ssSetOutputPortComplexSignal(S, 0, COMPLEX_NO);
             }
             else if(oPortComplexSignal == COMPLEX_YES)
             {
                   ssSetErrorStatus(S, "Output port must be real.");
          }
       }
    }
}
```

```
# define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL
static void mdlSetOutputPortComplexSignal(SimStruct *S, int_T port,
                                          int_T  oPortComplexSignal)
{
    /* Set the complex signal of the output ports */
    ssSetOutputPortComplexSignal(S, 0, oPortComplexSignal);

    if(oPortComplexSignal == COMPLEX_NO)
       {
        /* All inputs must be real */
        int_T i;

        for (i = 0; i < ENTREES; i++)
             {
            int_T iPortComplexSignal = ssGetInputPortComplexSignal(S, i);
            if(iPortComplexSignal == COMPLEX_INHERITED)
                    {
                ssSetInputPortComplexSignal(S, i, COMPLEX_NO);
            }
                    else if(iPortComplexSignal == COMPLEX_YES)
                    {
                           ssSetErrorStatus(S, "The output port is a 'real'signal.
"
                                  "All input ports must be 'real' signals.");
            }
        }
    }
       else
       {
             /* Output port is a complex signal.  Report an error, if all
         * inputs are real. */
             int_T i;
         boolean_T realInputs = true;
         for (i = 0; i < SORTIES; i++)
             {
            if(ssGetInputPortComplexSignal(S, i) != COMPLEX_NO)
                    {
                           realInputs = false;
                           break;
                    }
             }
             if(realInputs)
             {
            ssSetErrorStatus(S, "Input port and output port complex signal "
                           "mismatch. All input ports are 'real' signal. "
                           "The output port must be a 'real' signal.");
        }
    }
}
#endif /* MATLAB_MEX_FILE */

/* Required S-function trailer */

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

C-4 Estimateur de la valeur moyenne de la tension du lien cc

```
/*-------------------------------------------------------------------------
 TITRE:              Udo.c (Version dSPACE)
 AUTEUR :     H. Joël Nanga Ndjana
 DATE: Automne 2003
 DESCRIPTION: Permet d'estimer la valeur moyenne à la
                          sortie d'un redresseur triphasé à l'aide
                          des fonctions de commutation.
 Hypothèses de calcul :   1- L'angle d'empiètement est négligé.
                          2- Les tensions d'alimentation ne sont pas distordues.
                          3- Les déphasages que peuvent subir les tensions
                             sont négligés.
                          4- Les interrupteurs sont considérés parfaits.

 Copyright (c) 2003 CPEE - LTE.
-------------------------------------------------------------------------*/

#define S_FUNCTION_NAME  Udo
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include <math.h>

#define NUM_PARAMS (0)
#define Edo 281.00

#define ENTREES 3
#define SORTIES 2


/*-----------------------------------------------------/
  Permet de configurer les dimensions des vecteurs d'E/S
  -----------------------------------------------------*/

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S,  NUM_PARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
       {
             return;
       }

       ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;  /* Entrées */
    ssSetInputPortWidth(S, 0, ENTREES);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S,1)) return;  /* Sorties */
    ssSetOutputPortWidth(S, 0, SORTIES);

    ssSetNumSampleTimes(S, 1);
}
```

```
/*-------------------------------------------------------/
  Permet de spécifier que le pas d'échantillonnage utilisé
  sera celui spécifié dans l'outil de compilation
  -------------------------------------------------------*/
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs ============================================*/
static void mdlOutputs(SimStruct *S, int_T tid)
{

      static double SQ3 =  1.73205080756888;
    static double Van=1.0, Vbn=1.0, Vcn=1.0,Vmoy=0.0,old=0.0;
      static double t1=0, t3=0, t5=0, t7=0, t9=0, t11=0;
      static double A=0.0, B=0.0, C=0.0;
      static int sag=0,period=0;

    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    int_T             width = ssGetOutputPortWidth(S,0);


      /* Amplitude des tensions triphasées */
      Van=*uPtrs[0];
      Vbn=*uPtrs[1];
      Vcn=*uPtrs[2];

      /*  limiteur a 169.70562V  */
      if(Van > 169.70562)
      {
          Van = 169.70562;
      }
      else if(Vbn > 169.70562)
      {
          Vbn = 169.70562;
      }
      else if(Vcn > 169.70562)
      {
          Vcn = 169.70562;
      }


      /* Définition des instants de commutation */
      t1=atan2((SQ3*Vcn),((2.0*Van)+Vcn));
      t3=atan2((SQ3*(Vcn+Vbn)),(Vcn-Vbn));
      t5=atan2((-SQ3*Vbn),(2.0*Van+Vbn)) + M_PI;
      t7=atan2((SQ3*Vcn),(2.0*Van+Vcn)) + M_PI;
      t9=atan2((SQ3*(Vcn+Vbn)),(Vcn-Vbn)) + M_PI;
      t11=atan2((-SQ3*Vbn),(2.0*Van+Vbn)) + (2.0*M_PI);


      A=Van*(cos(t1)-cos(t5)-cos(t7)+cos(t11));
      B=-0.5*Vbn*(cos(t3)+cos(t5)-cos(t9)-cos(t11) - SQ3*(sin(t3)+sin(t5)-
sin(t9)-sin(t11)));
      C=0.5*Vcn*(cos(t1)+cos(t3)-cos(t7)-cos(t9) + SQ3*(sin(t1)+sin(t3)-sin(t7)-
sin(t9)));

      y[0] = (1.0/(2.0*M_PI))*(A+B+C); /* Valeur moyenne estimée */

      y[1] = Edo-y[0];
```

```
        if(y[1]<0.0) /* Limiteur de valeurs négatives */
        {
              y[1] = 0.0;
        }
}

static void mdlTerminate(SimStruct *S)
{
}

#if defined(MATLAB_MEX_FILE)
static void mdlSetInputPortComplexSignal(SimStruct *S, int_T port, int_T
iPortComplexSignal)
{
    int_T oPortComplexSignal = ssGetOutputPortComplexSignal(S,0);

    /* Set the complex signal of the input ports */
    ssSetInputPortComplexSignal(S, port, iPortComplexSignal);

    if(iPortComplexSignal == COMPLEX_YES)
       {
             /* Output port must be a complex signal */
        if(oPortComplexSignal == COMPLEX_INHERITED)
             {
            ssSetOutputPortComplexSignal(S, 0, COMPLEX_YES);
        }
             else if(oPortComplexSignal == COMPLEX_NO)
             {
                   ssSetErrorStatus(S, "Output port must be complex.");
        }
    }
       else if(oPortComplexSignal != COMPLEX_NO)
       {
       /*
        * The current input port is a real signal.  If the other input port
        * is a real signal, the output port must be a real signal.
        */
       int_T otherPort = (port == 0)? 1 : 0;
       int_T otherPortComplexSignal = ssGetInputPortComplexSignal(S, otherPort);
       if(otherPortComplexSignal == COMPLEX_NO)
       {
             /* Both input ports are real signals */
             if(oPortComplexSignal == COMPLEX_INHERITED)
             {
                   ssSetOutputPortComplexSignal(S, 0, COMPLEX_NO);
             }
             else if(oPortComplexSignal == COMPLEX_YES)
             {
                   ssSetErrorStatus(S, "Output port must be real.");
          }
       }
    }
}


# define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL
static void mdlSetOutputPortComplexSignal(SimStruct *S, int_T port,
                                    int_T  oPortComplexSignal)
{
    /* Set the complex signal of the output ports */
    ssSetOutputPortComplexSignal(S, 0, oPortComplexSignal);

    if(oPortComplexSignal == COMPLEX_NO)
```

```
        {
         /* All inputs must be real */
         int_T i;

         for (i = 0; i < ENTREES; i++)
              {
             int_T iPortComplexSignal = ssGetInputPortComplexSignal(S, i);
             if(iPortComplexSignal == COMPLEX_INHERITED)
                    {
                 ssSetInputPortComplexSignal(S, i, COMPLEX_NO);
             }
                    else if(iPortComplexSignal == COMPLEX_YES)
                    {
                           ssSetErrorStatus(S, "The output port is a 'real'
signal. "
                                  "All input ports must be 'real' signals.");
             }
         }
     }
        else
        {
              /* Output port is a complex signal.  Report an error, if all
          * inputs are real. */
              int_T i;
         boolean_T realInputs = true;
         for (i = 0; i < SORTIES; i++)
              {
             if(ssGetInputPortComplexSignal(S, i) != COMPLEX_NO)
                    {
                           realInputs = false;
                           break;
                    }
              }
              if(realInputs)
              {
             ssSetErrorStatus(S, "Input port and output port complex signal "
                           "mismatch. All input ports are 'real' signal. "
                           "The output port must be a 'real' signal.");
         }
     }
}
#endif /* MATLAB_MEX_FILE */

/* Required S-function trailer */

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

ANNEXE D : RÉALISATION D'UNE APPLICATION EMBARQUÉE À L'AIDE DU TMS320LF2407 (version eZdsp)

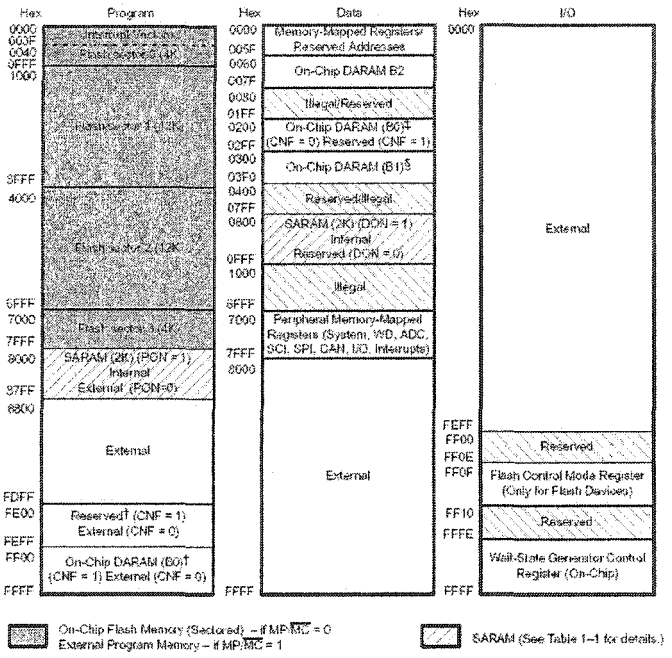
D-1 Introduction

Cette annexe traite de la façon de bâtir une application embarquée. C'est une application construite autour du DSP mais dont le fonctionnement ne dépend pas d'un autre système (un autre ordinateur ou de l'émulateur comme c'est le cas lors du développement). C'est une application de type *produit fini.* Pour construire ce type d'application, nous avons besoin de sauvegarder le programme en permanence (dans une ROM) dans le DSP. Le TMS320LF2407A dispose d'une mémoire flash (Flash EEPROM) qui nous permet de rencontrer cette exigence car elle est non-volatile. Elle est reprogrammable. La mémoire flash du TMS320LF2407A a une capacité de 32K mots de 16bits. Cette mémoire flash est subdivisée en quatre secteurs pouvant être individuellement protégés pendant l'effacement ou la programmation.

D-2 Particularités d'un programme embarqué

Globalement, un projet embarqué possède les mêmes constituantes qu'un projet normal; à savoir : le fichier contenant le programme en C et/ou assembleur (.c et/ou asm.), le fichier des vecteurs d'interruptions (.asm exemple : vecteur.asm), le fichier *command* (.cmd) et les fichiers en-tête (.h). Ces fichiers s'écrivent de la même façon qu'un programme normal à l'exception d'un : le fichier *command.*

Le fichier *command* contient les commandes d'édition de lien du programme et définit les différentes sections de la mémoire de DSP qui sont utilisées lors du chargement du programme en mémoire. Dans le cas d'une application embarquée, il faut modifié la définition de la mémoire à utiliser. Pour une application normale, on utilise généralement la mémoire vive interne.



NOTE A: Boot ROM: If the boot ROM is enabled, then address 0000–00FF in the program space will be occupied by boot ROM.
† When CNF = 1, addresses FE00h–FEFFh and FF00h–FFFFh are mapped to the same physical block (B0) in program-memory space. For example, a write to FE00h has the same effect as a write to FF00h. For simplicity, addresses FE00h–FEFFh are referred to as reserved when CNF = 1.
‡ When CNF = 0, addresses 0100h–01FFh and 0200h–02FFh are mapped to the same physical block (B0) in data-memory space. For example, a write to 0100h has the same effect as a write to 0200h. For simplicity, addresses 0100h–01FFh are referred to as reserved.
§ Addresses 0300h–03FFh and 0400h–04FFh are mapped to the same physical block (B1) in data-memory space. For example, a write to 0400h has the same effect as a write to 0300h. For simplicity, addresses 0400h–04FFh are referred to as reserved.

Figure D.1 : Mémoire du TMS320LF2407A [36]

D-3 Programmation de la mémoire flash

L'effacement et la programmation (l'exécution des programmes d'effacement et de programmation) de la mémoire flash sont effectués par l'unité centrale du DSP.

Pour programmer la mémoire flash (interne) du TMS320LF2407x, il existe une application possédant une interface graphique sur le site web de Texas instrument.

Cette application est compatible avec Code Composer et s'y intègre bien; puisqu'elle a besoin Code Composer lors du téléchargement du programme dans la mémoire flash via un port JTAG ou le port parallèle. Cette application se nomme ***TMS320LF240x Flash Programmer Plug-In 1.1.2.***

Le fichier à télécharger est *c2000flashprogsw_v112.zip*. Après l'avoir décompressé, il faut l'installer à l'aide du fichier exécutable *setup*.

Une fois l'installation terminée, couper l'alimentation de la carte pour effectuer le changement de mode du DSP. Cette dernière opération se fait en plaçant les jumpers JP3 et JP4 de la carte DSP à la position 2-3 [36].

Rétablir l'alimentation du DSP.

NOTE : Avant de commencer la programmation de la mémoire flash, il est important de faire un RESET du DSP.

Démarrer Code Composer et lancer le nouvel outil à partir du menu *TOOLS* de Code composer (Figure D.2).

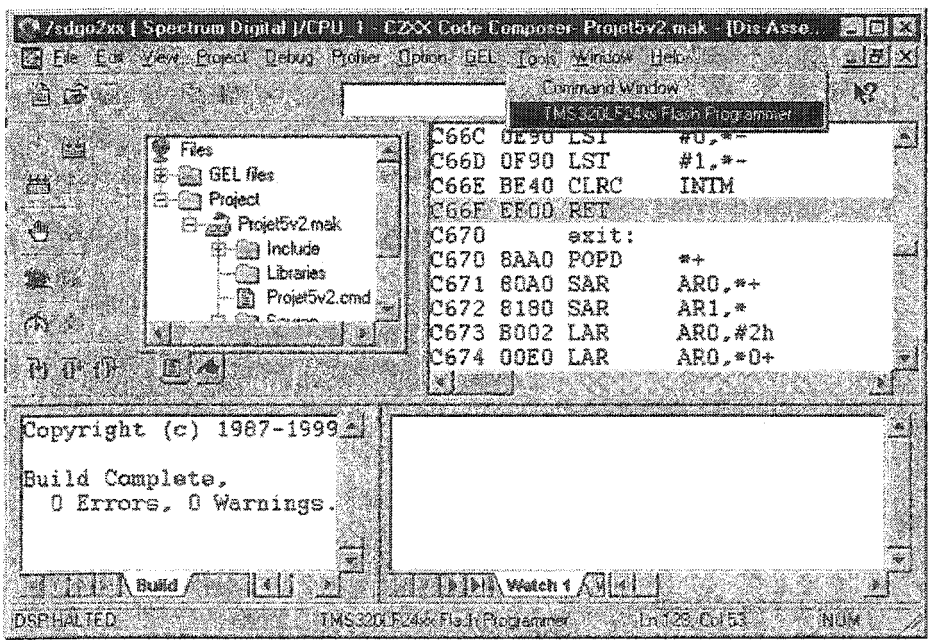


Figure D.2 : Lancement du programmeur de la mémoire Flash

Cochez AutoDetect Device Family (Default). L'application devrait reconnaître votre carte et afficher son nom.

Ensuite, appuyer sur *Clear* et ensuite sur *Erase* pour effacer la mémoire Flash. Entre chaque action il est important de vérifier si l'action demandée a été réussie. Ce résultat est donné par *STATUS*. Pour finir la programmation de la mémoire flash proprement dite, appuyer sur *Program Code Only*.

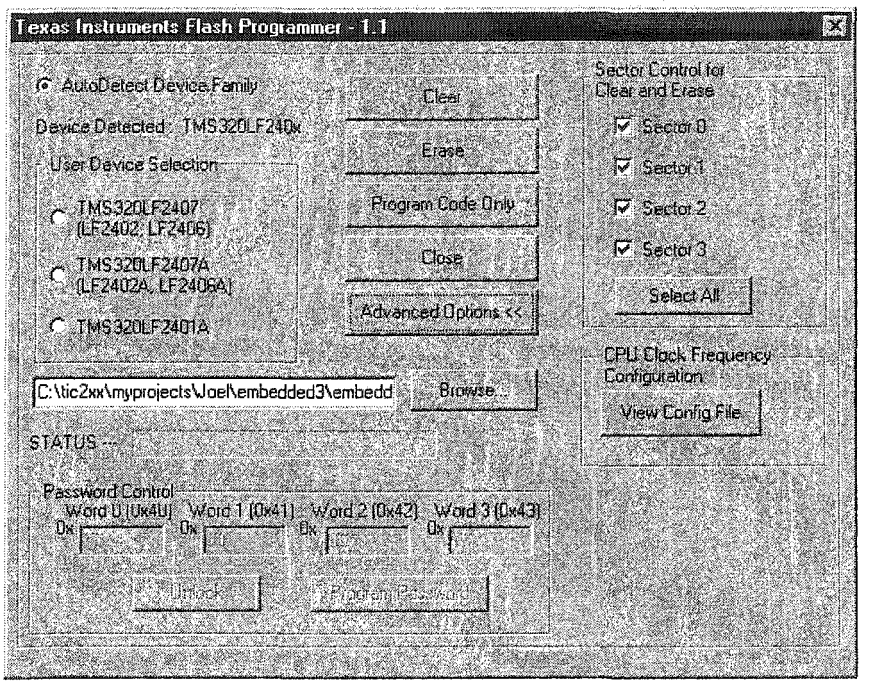


Figure D.3 : Programmation de la mémoire Flash

Une fois la mémoire flash programmée, fermer Code Composer et couper de nouveau l'alimentation du DSP. Remettre le jumper JP3 à la position 1-2 [36]. Aussitôt, en rétablissant l'alimentation du DSP, il commencera automatiquement l'exécution du programme se trouvant dans la mémoire flash.

ANNEXE E : PROGRAMMES DSP DE LA COMMANDE DU REDRESSEUR TRIPHASÉ À THYRISTORS

Cette annexe présente les programmes DSP pour la commande d'un redresseur triphasé à thyristors.

E-1 Programmes sources

Les programmes sources sont écrits en langage C avec quelques instructions en assembleur.

E-1-1 Programme source du premier DSP

```
/*-------------------------------------------------------------
  NOM DU FICHIER :  ProjetSCR.c
  DATE              :     Octobre 2003
  AUTEUR          :  Hervé Joël Nanga Ndjana

  DESCRIPTION : Carte de commande pour SCR - Carte #1

  ============ CONVERSION ANALOGIQUE NUMERIQUE ===============

                          ADCIN0 : Permet de fixer l'angle d'amorcage
                                   (0V ----> 0 degré
                                    3.3V ----> 125 degré
                                    9.457V ----> 360 degré)

  ============  CAPTURE DES SIGNAUX ==========================
                          CAP1    : 1er signal de synchronisation (Vac)
                          CAP2    : 2e signal de synchronisation (Vba)
 /*-------------------------------------------------------------*/

#include "regs2407.h"

#define      V_MIN  0.025        /* Tension min. correspondant à l'angle min.  */
#define      V_MAX  9.457        /* Tension max. correspondant à l'angle max.  */

#define      ALPHA_MIN    0      /* Angle minimum */
#define      ALPHA_MAX    360    /* Angle maximum */

#define      PW     10000        /* Durée de l'impulsion    */
#define      PERIOD_MAX   41500
#define      DEMI_T 20000  /* Délai entre 2 pairs d'impulsions*/
#define      RAZ    0xFFFF
#define      LENGTH 10
```

```
/* vecteurs d'interruptions */
interrupt void ADC_ISR(void);
interrupt void CAP_ISR(void);

void config_CPU(void);
void config_ADC(void);
void config_CAPTURE(void);
void config_PWM(void);
void Start_Timer(void);

unsigned int PERIOD_B;
double DT=0.5, TON_B;

double a=0.0, b=0.0;
double tab[LENGTH]={0.0};
double Alpha, Alpha_p, Vcom, Vsum;
unsigned int PERIOD_A;
int adc=0, j=0;


void main(void)
{
      /*============= Initialisation des variables ===============*/

      PERIOD_B = PERIOD_MAX;     /* 60Hz = 41500 */


    /* Alpha = a*Vcom + b */
    a = (ALPHA_MAX - ALPHA_MIN)/(V_MAX - V_MIN);
    b = ALPHA_MAX - a*V_MAX;

    asm (" setc INTM");          /* Interdire toutes les interruptions */

    config_CPU();                /* Initialisation du CPU                */
    config_ADC();          /* Initialisation de l'unité ADC       */
    config_CAPTURE();            /* Initialisation de l'unité de capture*/


    config_PWM();          /* initialisation des circuits PWM */

    asm (" clrc INTM"); /* Permettre toutes les interruptions non-masquables */


    Start_Timer(); /* Démarrer toutes les operations, donc
                      les operations PMW, ADC et de Capture */
      while(1)
      {

        Vsum=0.0;
        for(j=0;j<LENGTH;j++)   /* On moyenne les LENGTH nieme valeurs */
        {                              /* de l'entrée de consigne */
                Vsum=Vsum+tab[j];
        }
        Vcom = Vsum/LENGTH;
        if(Vcom < V_MIN)
        {
             Vcom = V_MIN;
        }
        /* Calcul de l'angle d'amorçage */
        Alpha = a*Vcom + b;
        Alpha_p = Alpha*((PERIOD_MAX -1)/ALPHA_MAX) + 1;
        PERIOD_A = (unsigned int)Alpha_p;
```

```
        /*--- thy1,4 ----*/
        T1CMPR = PERIOD_A;                      /* T1CMPR */

        CMPR1 = PERIOD_A + PW;                  /* PWM 1,2 */
        CMPR2 = DEMI_T + PERIOD_A;              /* PWM 3,4 */
        CMPR3 = CMPR2 + PW;                     /* PWM 5,6 */

        /*--- thy3,6 ----*/
        T3CMPR = T1CMPR;                /* T3CMPR */

        CMPR4 = CMPR1;                   /* PWM 7,8    */
        CMPR5 = CMPR2;                   /* PWM 9,10 */
        CMPR6 = CMPR3;                          /* PWM 11,12 */
    }
}


void config_CPU(void)
{
    asm (" clrc SXM");  /* Clear Sign Extension Mode bit */
    asm (" clrc OVM");  /* Reset Overflow Mode bit       */
    asm (" clrc CNF");  /* Configure block B0 to data mem.     */

    WSGR=0x0000;        /* set the external waitstates WSGR*/

    WDCR=0x006F;
    /* Initialize Watchdog-timer Control register
    6:      1: disabled
    5:      1: Normal operation
    4:      0 : Normal Operation
    3:   1: Normal Operation
    2-0: 111: Watchdog prescaler 7 : div 64*/

    SCSR1= 0x0E8D;
    /*-------  SETUP for the SCSR1 - Register
    CLKSRC        0 : intern(40MHz)
    LPM1,0        00 : Low power mode 0 if idle
    CLK_PS        111 : PLL multiply by 0.5
    ADC_CLKEN           1 : Use ADC-service
    SCI_CLKEN           0 : No SCI-service
    SPI_CLKEN           0 : No SPI-servide
    CAN_CLKEN           0 : No CAN-service
    EVB_CLKEN           1 : Use EVB-Service
    EVA_CLKEN           1 : Use EVA-Service
    ILLADR        1 : Clear ILLADR during startup  */

    /*----- Setup the I/O pins ----*/

    MCRA = 0xBFF8;
    MCRB = 0xFE00;
    MCRC = 0x3C7F;


  /*----- Setup the event manager A interrupts ----*/
    EVAIFRA = 0xFFFF;   /* clear all EVA group A interrupts */
    EVAIFRB = 0xFFFF;   /* clear all EVA group B interrupts */
    EVAIFRC = 0xFFFF;   /* clear all EVA group C interrupts */

    EVAIMRA = 0x0000;   /* No interrupts in this groups is allowed */
    EVAIMRB = 0x0000;   /* No interrupts in this groups is allowed */
    EVAIMRC = 0x0003;   /* Allows CAP1INT and CAP2INT */
```

```
        /*----- Setup the event manager B interrupts ----*/
        EVBIFRA = 0xFFFF;   /* clear all EVA group A interrupts */
        EVBIFRB = 0xFFFF;   /* clear all EVA group B interrupts */
        EVBIFRC = 0xFFFF;   /* clear all EVA group C interrupts */

        EVBIMRA = 0x0000;   /* No interrupts in this groups is allowed*/
        EVBIMRB = 0x0000;   /* No interrupts in this groups is allowed*/
        EVBIMRC = 0x0000;   /* No interrupts in this groups is allowed*/

        IMR = 0x0000;       /* clear the IMR register       */
        IFR = 0xFFFF;       /* Reset  all core interrupts */
        IMR = 0x0009;       /* enable desired core interrupts
                                          (INT1 and INT4)    */

        GPTCONA = 0x644A;
        GPTCONB = 0x644A;
        /*------  SETUP for the GPTCONA - Register ----
        10-9 : 10    ADC-Start by GPT2-Period-Event
        8-7  : 00   no ADC-Start by GPT1-Event
        6       : 1    Enable all GPT compare outputs
        3-2  : 01   Pol. of GPT2 comp out= Active high
        1-0  : 01   Pol. of GPT1 comp out= Active high  */
}

void config_CAPTURE(void)
{
        /* Set up the capture pins to primary functions
           see config_CPU() */

    CAPCONA = 0xB054;    /* Set up capture units*/
    CAPFIFOA = 0x0000;       /* No entry*/

    CAPCONB = 0x0000;     /* disable EVB capture unit */
    CAPFIFOB = 0x0000;
}

void config_PWM(void)
{

        COMCONA = 0x8200;
        COMCONB = 0x8200;

        DBTCONA = 0x0000;          /* No dead band*/
        DBTCONB = 0x0000;

        T1PR = 0xFDE8;
        T1CNT = 0x0000;

        T2PR = 0x0320;
        T2CNT = 0x0000;

        T3PR = 0xFDE8;
        T3CNT = 0x0000;

        T4PR = 0xFDE8;
        T4CNT = 0x0000;
}

void Start_Timer(void)
{

    ACTRA = 0x0555;
    ACTRB = 0x0555;  /* all active low */
```

```
    T1CON = 0x91C6;       /* Timer1 start */
    T3CON = 0x91C6;       /* Timer3 start */
    T2CON = 0x90C6;       /* Timer2 start */
    T4CON = 0x90C6;       /* Timer4 start */
    /*-------- Initialisation des registres TxCON --------
       15-14 FREE, SOFT : 10      No emulation mode
       12-11 : TMODE1,0 : 10      continuous up-counting
       10-8  : TPS2-0   : 000     Input clock prescaler CPUCLK/1

       6     : TENABLE  : 1       Enable GPT1

       5-4   : TCLKS1,0 : 00      Clock source select : internal

       3-2   : TCLD1,0  : 01      Timer compare(active) register reload condition
                            when counter value is 0 or equal to period register
       1     :  TECMPR      : 0   Disable timer compare operation
       0     :      SELT1PR       : 0    use own per. reg. */
}

void config_ADC(void)
{
       CALIBRATION = 0;           /* ADC-offset is not corrected   */
       ADCTRL1 = 0x4000;          /* Reset on the entire ADC-module (bit14 = 1)*/
       asm (" nop");            /* delay to allow a good initialisation */
       ADCTRL1 = 0x3700;
       /*---  SETUP for the ADCTRL1 - Register ----
       RESET                14    : 0 No effect
       SOFTFREE             13-12: 11 free run
       ACQ_PS               11-8 : Acquisition time 16 x TClk
       CPS                        7    : 0 ADC logic Clock = CLK/1
       CONT_RUN         6     : 0 Stop after each sequence
       INT_PRI                    5    : 0 High ADC interrupt priority
       SEQ_CASC             4     : 0 Dual-Sequencer mode
       CALIBRATION          3-0   : 0 Calibration and self-test function */

       CHSELSEQ1 = 0x0000;        /* Channel 0 is scanned */

       ADCTRL2 |= 0x5000;  /* immediate reset or start calibration if bit3 of
       ADCTRL = 1                    SEQ1_BSY = 1 : in progress      */
       ADCTRL2 = 0x95D1;
       /*-------   SETUP for the ADCTRL2 - Register -----
       EVB_SOC_SEQ          15 : 1 EVB starts Cascaded Sequ.
       RST_SEQ1             14 : 1 Reset Sequencer 1
       SOC_SEQ1             13 : 0 Clears a pending SOC trig
       INT_ENA_SEQ1   11-10 : 1 Interrupt Mode 1
       EVA_SOC_SEQ1   8: 1 EVA starts Sequencer1
       EXT_SOC_SEQ1   7: 1 ADCSOC Pin starts Sequencer1
       RST_SEQ2             6 : 1 Reset Sequencer 2
       SOC_SEQ2             5 : 0 Clears a pending SOC trig
       INT_ENA_SEQ2   3-2 : 0 Interrupt SEQ2 disabled
       EVB_SOC_SEQ2      0: 1 EVB starts Sequencer1   */
       MAXCONV = 0;
}


interrupt void ADC_ISR(void)
{
       if((PIVR-0x0004)==0)       /* Verify type of interrupt ( 4 = ADC ) */
       {
              tab[adc] = ((double)(RESULT0>>6)/1023.0)*3.33;
              adc++;
              if(adc>(LENGTH-1))
```

```
            {
                  adc=0;
            }
            ADCTRL2 = ADCTRL2 | 0x0000;      /* clear ADC-Sequencer1-Interrupt
flag(bit9) and reset Sequencer (bit14) */
      }
}

interrupt void CAP_ISR(void)
{

      if(((PIVR-0x0034)==0))
      {
            /* CAP2 */
            EVAIFRC = EVAIFRC & 0x0002;      /* Reset du drapeau de CAP2INT */
            T3CNT = RAZ;
      }
      else if(((PIVR-0x0033)==0))
      {
            /* CAP1 */
            EVAIFRC = EVAIFRC & 0x0001; /* Reset du drapeau de CAP1INT */
            T1CNT = RAZ;
      }
}
```

E-1-2 Programme source du second DSP

```
/*-------------------------------------------------------------
  NOM DU FICHIER :  ProjetSCR2.c
  DATE              :     Octobre 2003
  AUTEUR          :  Hervé Joël Nanga Ndjana

  DESCRIPTION : Carte de commande pour SCR - Carte #2

  ============ CONVERSION ANALOGIQUE NUMERIQUE ================


                        ADCIN0 : Permet de fixer l'angle d'amorcage
                                 (0V ----> 0 degré
                                  3.3V ----> 125 degré
                                  9.457V ----> 360 degré)

  ============  CAPTURE DES SIGNAUX ==========================

                        CAP2    : 3e signal de synchronisation (Vcb)
/*-------------------------------------------------------------*/

#include "regs2407.h"

#define      V_MIN  0.025        /* Tension min. correspondant à l'angle min.  */
#define      V_MAX  9.457        /* Tension max. correspondant à l'angle max.  */


#define      ALPHA_MIN    0      /* Angle minimum */
#define      ALPHA_MAX    360    /* Angle maximum */

#define      PW     10000        /* Durée de l'impulsion 90 degre =6700 */
#define      PERIOD_MAX   41500
#define      DEMI_T 20000  /* Délai entre 2 pairs d'impulsions*/
#define      RAZ    0xFFFF
```

```
#define      LENGTH 10

/* Vecteurs d'interruptions */
interrupt void ADC_ISR(void);
interrupt void CAP_ISR(void);


void config_CPU(void);
void config_ADC(void);
void config_CAPTURE(void);
void config_PWM(unsigned int, unsigned int);
void Start_Timer(void);


unsigned int PERIOD_B, PERIOD_H;
double DT=0.5, TON_B, TON_H;

double a=0.0, b=0.0;
double tab[LENGTH]={0.0};

double Alpha, Alpha_p, Vcom, Vsum;
unsigned int PERIOD_A;
int adc=0,j=0;

void main(void)
{
      /*============== Initialisation des variables ===============*/

      PERIOD_B = PERIOD_MAX;     /* 60Hz */
      PERIOD_H = 300;                  /* 300 = 16.5kHz */
      TON_H = 0.8 * PERIOD_H;


      /* Alpha = a*Vcom + b */
      a = (ALPHA_MAX - ALPHA_MIN)/(V_MAX - V_MIN);
      b = ALPHA_MAX - a*V_MAX;


      asm (" setc INTM");        /* Interdire toutes les interruptions */

      config_CPU();              /* Initialisation du CPU */
      config_ADC();                    /* Initialisation de l'unité ADC       */
      config_CAPTURE();          /* Initialisation de l'unité de capture*/


      config_PWM(PERIOD_H, TON_H);/* Initialisation des circuits PWM */

      asm (" clrc INTM");/*Permettre toutes les interruptions non-masquables */

      Start_Timer();             /* Démarrer toutes les opérations, donc
                                 les opérations PMW, ADC et de Capture */
      while(1)
      {
         Vsum=0.0;
         for(j=0;j<LENGTH;j++)   /* On moyenne les LENGTH nieme valeurs */
         {                              /* de l'entrée de consigne */
                  Vsum=Vsum+tab[j];
         }
         Vcom = Vsum/(double)LENGTH;
         if(Vcom < V_MIN)              /* On fixe l'angle minimum */
         {
               Vcom = V_MIN;
         }
```

```
            Alpha = a*Vcom + b;     /* Calcul de l'angle d'amorçage */
            Alpha_p = Alpha*((PERIOD_MAX -1)/ALPHA_MAX) + 1;
            PERIOD_A = (unsigned int)Alpha_p;

            /*--- thy5 & thy2 ----*/
            T1CMPR = PERIOD_A;                  /* T1CMP */

            CMPR1 = PERIOD_A + PW;              /* PWM 1,2 */
            CMPR2 = DEMI_T + PERIOD_A;    /* PWM 3,4 */
            CMPR3 = CMPR2 + PW;                 /* PWM 5,6 */
        }
    }


void config_CPU(void)
{
        asm (" clrc SXM");          /* Clear Sign Extension Mode bit */
        asm (" clrc OVM");          /* Reset Overflow Mode bit       */
        asm (" clrc CNF");          /* Configure block B0 to data mem.      */

        WSGR=0x0000; /* set the external waitstates    WSGR */

        WDCR=0x006F;
        /* Initialize Watchdog-timer Control register
        6:      1: disabled
        5:      1: Normal operation
        4:      0 : Normal Operation
        3:   1: Normal Operation
        2-0: 111: Watchdog prescaler 7 : div 64 */

        SCSR1= 0x0E8D;
        /*-------  SETUP for the SCSR1 - Register
        CLKSRC              0 : intern(40MHz)
        LPM1,0              00 : Low power mode 0 if idle
        CLK_PS              111 : PLL multiply by 0.5
        ADC_CLKEN           1 : Use ADC-service
        SCI_CLKEN         0 : No SCI-service
        SPI_CLKEN           0 : No SPI-servide
        CAN_CLKEN           0 : No CAN-service
        EVB_CLKEN           1 : Use EVB-Service
        EVA_CLKEN           1 : Use EVA-Service
        ILLADR              1 : Clear ILLADR during startup  */

        /*----- Setup the I/O pins ----*/

        MCRA = 0xBFF8;
        MCRB = 0xFE00;
        MCRC = 0x3C7F;


    /*----- Setup the event manager A interrupts ----*/
        EVAIFRA = 0xFFFF;   /* clear all EVA group A interrupts */
        EVAIFRB = 0xFFFF;   /* clear all EVA group B interrupts */
        EVAIFRC = 0xFFFF;   /* clear all EVA group C interrupts */

        EVAIMRA = 0x0000;   /* No interrupts in this groups is allowed */
        EVAIMRB = 0x0000;   /* No interrupts in this groups is allowed */
        EVAIMRC = 0x0003;   /* Allows CAP1INT and CAP2INT */

        /*----- Setup the event manager B interrupts ----*/
        EVBIFRA = 0xFFFF;   /* clear all EVA group A interrupts */
        EVBIFRB = 0xFFFF;   /* clear all EVA group B interrupts */
        EVBIFRC = 0xFFFF;   /* clear all EVA group C interrupts */
```

```
        EVBIMRA = 0x0000;   /*  No interrupts in this groups is allowed */
        EVBIMRB = 0x0000;   /* No interrupts in this groups is allowed */
        EVBIMRC = 0x0000;   /* No interrupts in this groups is allowed */

        IMR = 0x0000;       /* clear the IMR register       */
        IFR = 0xFFFF;       /* Reset  all core interrupts */
        IMR = 0x0009;       /* enable desired core interrupts
                                    (INT1 and INT4)     */

        GPTCONA = 0x644A;
        GPTCONB = 0x644A;
        /*------  SETUP for the GPTCONA - Register ----
        10-9 : 10    ADC-Start by GPT2-Period-Event
        8-7  : 00   no ADC-Start by GPT1-Event
        6       : 1    Enable all GPT compare outputs
        3-2  : 01   Pol. of GPT2 comp out= Active high
        1-0  : 01   Pol. of GPT1 comp out= Active high  */
}

void config_CAPTURE(void)
{
        /* Set up the capture pins to primary functions
           see config_CPU() */

        CAPCONA = 0xA050;       /* Set up capture units*/
        CAPFIFOA = 0x0000;  /* CAP2FIFO has no entry*/

        CAPCONB = 0x0000;       /* disable EVB capture unit */
        CAPFIFOB = 0x0000;
}

void config_PWM(unsigned int PERIOD_H, unsigned int TON_H)
{


        COMCONA = 0x8200;
        COMCONB = 0x8200;

        DBTCONA = 0x0000;         /* No dead band*/
        DBTCONB = 0x0000;

        T1PR = 0xFDE8;
        T1CNT = 0x0000;
        T2PR = 0x0320;
        T2CNT = 0x0000;
        T3PR = PERIOD_H;
        T3CMPR = TON_H;
        T3CNT = 0x0000;
        T4PR = 0xFDE8;
        T4CNT = 0x0000;
}

void Start_Timer(void)
{

        ACTRA = 0x0555;
        ACTRB = 0x0555;      /* all active low */
        T1CON = 0x91C6;   /* Timer1 start */
        T3CON = 0x90C6;   /* Timer3 start */
        T2CON = 0x90C6;   /* Timer2 start */
        T4CON = 0x90C6;   /* Timer4 start */
        /*------- Initialisation des registres TxCON --------
```

```
        15-14 FREE, SOFT : 10       No emulation mode
        12-11 : TMODE1,0 : 10       continuous up-counting
        10-8  : TPS2-0   : 000      Input clock prescaler CPUCLK/1

        6     : TENABLE  : 1        Enable GPT1

        5-4   : TCLKS1,0 : 00       Clock source select : internal

        3-2   : TCLD1,0  : 01       Timer compare(active) register reload condition
                                when counter value is 0 or equal to period register
        1     :  TECMPR      : 0    Disable timer compare operation
        0       :    SELT1PR       : 0    use own per. reg. */
}


void config_ADC(void)
{
        CALIBRATION = 0;    /* ADC-offset is not corrected */

        ADCTRL1 = 0x4000;          /* Reset on the entire ADC-module (bit14 = 1)*/
        asm (" nop");            /* delay to allow a good initialisation */
        ADCTRL1 = 0x3700;
        /*---  SETUP for the ADCTRL1 - Register ----
        RESET                   14    : 0 No effect
        SOFTFREE                13-12: 11 free run
        ACQ_PS                  11-8  : Acquisition time 16 x TClk
        CPS                           7     : 0 ADC logic Clock = CLK/1
        CONT_RUN          6     : 0 Stop after each sequence
        INT_PRI                       5     : 0 High ADC interrupt priority
        SEQ_CASC                4     : 0 Dual-Sequencer mode
        CALIBRATION             3-0   : 0 Calibration and self-test function */

        CHSELSEQ1 = 0x0000;         /* Channel 0 is scanned */

        ADCTRL2 |= 0x5000;          /* immediate reset or start calibration if bit3
                                    of ADCTRL = 1  SEQ1_BSY = 1 : in progress      */
        ADCTRL2 = 0x95D1;
        /*-------   SETUP for the ADCTRL2 - Register -----
        EVB_SOC_SEQ             15 : 1 EVB starts Cascaded Sequ.
        RST_SEQ1                14 : 1 Reset Sequencer 1
        SOC_SEQ1                13 : 0 Clears a pending SOC trig
        INT_ENA_SEQ1   11-10 : 1 Interrupt Mode 1
        EVA_SOC_SEQ1   8: 1 EVA starts Sequencer1
        EXT_SOC_SEQ1   7: 1 ADCSOC Pin starts Sequencer1
        RST_SEQ2                6 : 1 Reset Sequencer 2
        SOC_SEQ2                5 : 0 Clears a pending SOC trig
        INT_ENA_SEQ2   3-2 : 0 Interrupt SEQ2 disabled
        EVB_SOC_SEQ2      0: 1 EVB starts Sequencer1   */
        MAXCONV = 0;
}


interrupt void ADC_ISR(void)
{

        if((PIVR-0x0004)==0)        /* Verify type of interrupt ( 4 = ADC ) */
        {
                tab[adc] = ((double)(RESULT0>>6)/1023.0)*3.33;
                adc++;
                if(adc>(LENGTH-1))
                {
                        adc=0;
                }
```

```
            ADCTRL2 = ADCTRL2 | 0x0000;      /* clear ADC-Sequencer1-Interrupt
            flag(bit9) and reset Sequencer (bit14) */
      }
}

interrupt void CAP_ISR(void)
{
    if(((PIVR-0x0034)==0))
      {
            /* CAP2 */
            EVAIFRC = EVAIFRC & 0x0002;      /* Reset du drapeau CAP2INT */
            T1CNT = RAZ;
      }

}
```

E-2 Fichier des vecteurs d'interruption

Le fichier des vecteurs d'interruption est le même pour les deux DSP.

```
/*-------------------------------------------------------------
  NOM DU FICHIER :  vectors.ASM
  DATE      :  Octobre 2003
  AUTEUR          :  Hervé Joël Nanga Ndjana

  DESCRIPTION : Table des vecteurs d'interruptions pour le TMS320LF240x
                pour les programmes en langage C
/*-------------------------------------------------------------*/
            .title "vectors.asm"

        .ref _c_int0, _ADC_ISR, _CAP_ISR

        .sect    ".vectors"
reset:  B       _c_int0         ;00h reset
int1:   B       _ADC_ISR        ;02h INT1
int2:   B       int2                ;04h INT2
int3:   B       int3                  ;06h INT3
int4:   B       _CAP_ISR        ;08h INT4
int5:   B       int5            ;0Ah INT5
int6:   B       int6            ;0Ch INT6
int7:   B       int7            ;0Eh reserved
int8:   B       int8            ;10h INT8  (software)
int9:   B       int9            ;12h INT9  (software)
int10:  B       int10           ;14h INT10 (software)
int11:  B       int11           ;16h INT11 (software)
int12:  B       int12           ;18h INT12 (software)
int13:  B       int13           ;1Ah INT13 (software)
int14:  B       int14           ;1Ch INT14 (software)
int15:  B       int15           ;1Eh INT15 (software)
int16:  B       int16           ;20h INT16 (software)
int17:  B       int17           ;22h TRAP
int18:  B       int18           ;24h NMI
int19:  B       int19           ;26h reserved
int20:  B       int20           ;28h INT20 (software)
int21:  B       int21           ;2Ah INT21 (software)
int22:  B       int22           ;2Ch INT22 (software)
int23:  B       int23           ;2Eh INT23 (software)
int24:  B       int24           ;30h INT24 (software)
int25:  B       int25           ;32h INT25 (software)
int26:  B       int26           ;34h INT26 (software)
int27:  B       int27           ;36h INT27 (software)
int28:  B       int28           ;38h INT28 (software)
```

```
int29:  B       int29           ;3Ah INT29 (software)
int30:  B       int30           ;3Ch INT30 (software)
int31:  B       int31           ;3Eh INT31 (software)
```

E-3 Fichier *command*

Comme dans le cas du fichier des vecteurs d'interruption, le fichier *command* est le même pour les deux DSP

```
/*-------------------------------------------------------
  NOM DU FICHIER :  ProjetSCR.cmd
  DESCRIPTION : C code linker command file for LF2407 DSP

-------------------------------------------------------*/
-c                          /* ROM autoinitialization */
-x                  /* force rereading libraries */
-o ProjetSCR.out            /* output file */
-m ProjetSCR.map            /* map file */
/* files to be linked */
ProjetSCR.obj
vectors.obj

-l rts2xx.lib    /* Run Time Support  */

  /* specify memory map */

MEMORY
{
  PAGE 0:    /* Program Memory */
    VECS:           org=00000h,   len=00040h   /* internal FLASH */
    FLASH:          org=00044h,   len=07FBCh   /* internal FLASH */
    EXTPROG:        org=08800h,   len=07800h   /* external SRAM  */

  PAGE 1:    /* Data Memory */
    B2:             org=00060h,   len=00020h   /* internal DARAM */
    B0:             org=00200h,   len=00100h   /* internal DARAM */
    B1:             org=00300h,   len=00100h   /* internal DARAM */
    SARAM:          org=00800h,   len=00800h   /* internal SARAM */
    EXTDATA:        org=08000h,   len=08000h   /* external SRAM  */
}

SECTIONS
{
/* Sections generated by the C-compiler */
        .text:  >   FLASH      PAGE 0   /* initialized */
       .cinit:  >   FLASH      PAGE 0   /* initialized */
       .const:  >   B1         PAGE 1   /* initialized */
      .switch:  >   FLASH      PAGE 0   /* initialized */
         .bss:  >   B1         PAGE 1   /* uninitialized */
       .stack:  >   SARAM      PAGE 1   /* uninitialized */
      .sysmem:  >   B1         PAGE 1   /* uninitialized */

/* Sections declared by the user */
      vectors:  >   VECS       PAGE 0   /* initialized */
}
```

```
/*-------------------------------------------------------
  NOM DU FICHIER :  ProjetSCR2.cmd
  DESCRIPTION : C code linker command file for LF2407 DSP

-------------------------------------------------------*/

-c                  /* ROM autoinitialization */
-x                 /* force rereading libraries */
-o ProjetSCR2.out      /* output file */
-m ProjetSCR2.map      /* map file */

/* files to be linked */
ProjetSCR2.obj
vectors.obj

-l rts2xx.lib     /* Run Time Support  */


  /* specify memory map */

MEMORY
{
  PAGE 0:     /* Program Memory */
    VECS:            org=00000h,   len=00040h   /* internal FLASH */
    FLASH:           org=00044h,   len=07FBCh   /* internal FLASH */
    EXTPROG:         org=08800h,   len=07800h   /* external SRAM  */

  PAGE 1:     /* Data Memory */
    B2:              org=00060h,   len=00020h   /* internal DARAM */
    B0:              org=00200h,   len=00100h   /* internal DARAM */
    B1:              org=00300h,   len=00100h   /* internal DARAM */
    SARAM:           org=00800h,   len=00800h   /* internal SARAM */
    EXTDATA:         org=08000h,   len=08000h   /* external SRAM  */
}

SECTIONS
{
/* Sections generated by the C-compiler */
        .text:  >   FLASH      PAGE 0   /* initialized */
       .cinit:  >   FLASH      PAGE 0   /* initialized */
       .const:  >   B1         PAGE 1   /* initialized */
      .switch:  >   FLASH      PAGE 0   /* initialized */
         .bss:  >   B1         PAGE 1   /* uninitialized */
       .stack:  >   SARAM      PAGE 1   /* uninitialized */
      .sysmem:  >   B1         PAGE 1   /* uninitialized */

/* Sections declared by the user */
      vectors:  >   VECS       PAGE 0   /* initialized */
}
```

ANNEXE F : SCHÉMA DU CIRCUIT ÉLECTRONIQUE DE COMMANDE

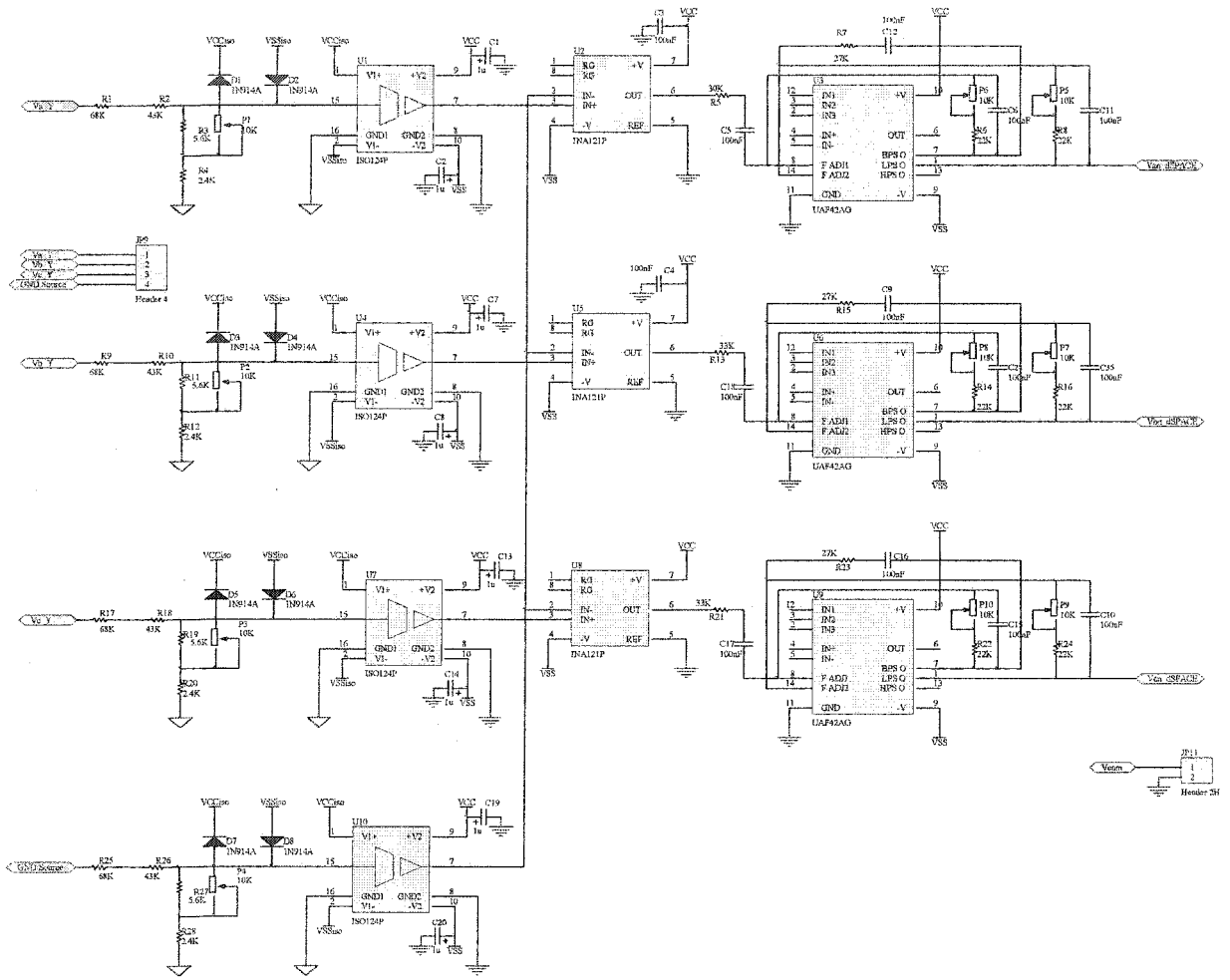


Figure F.1 : Mesure des tensions de ligne

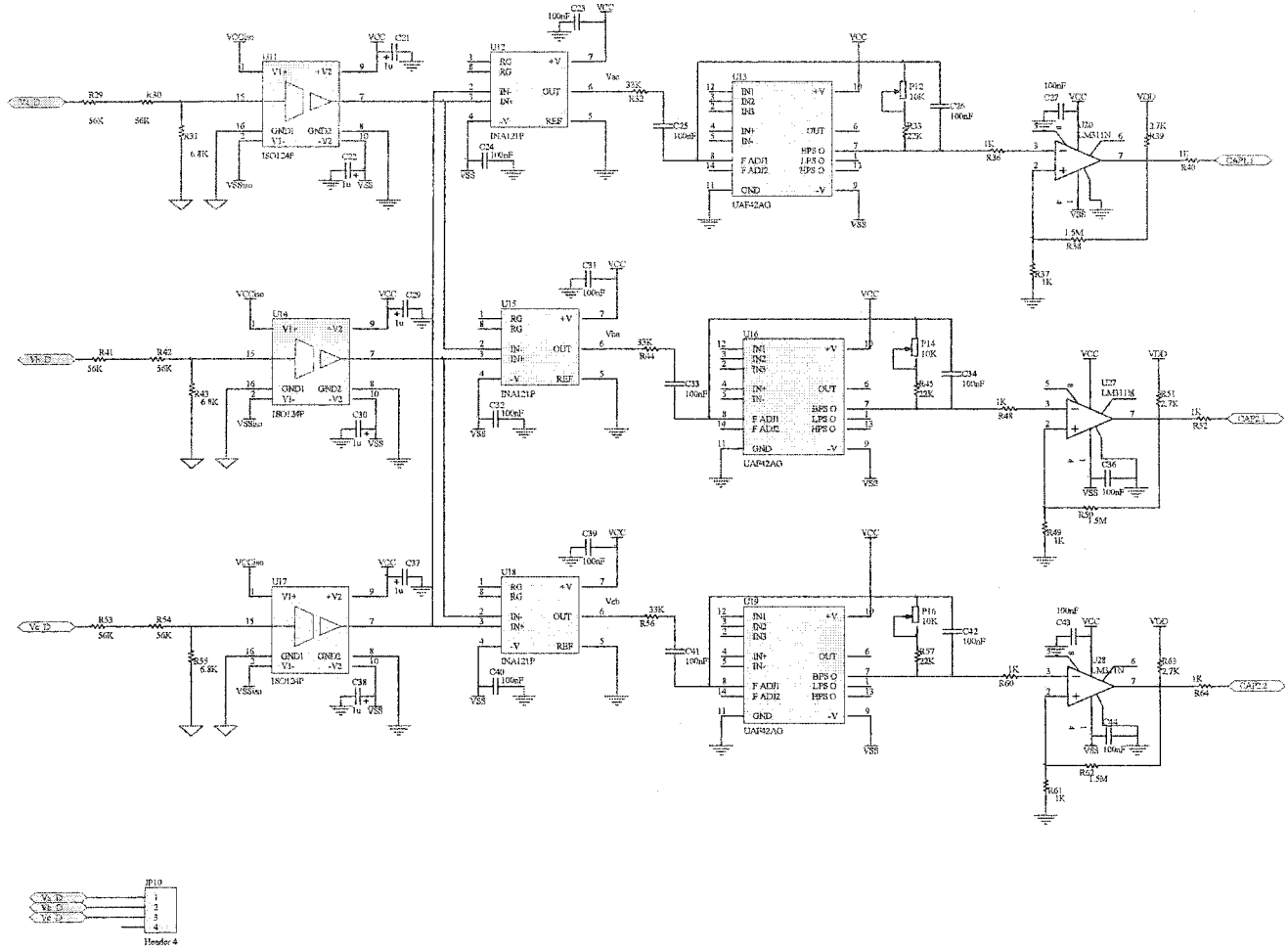


Figure F.2 : Circuit de génération des signaux de synchronisation

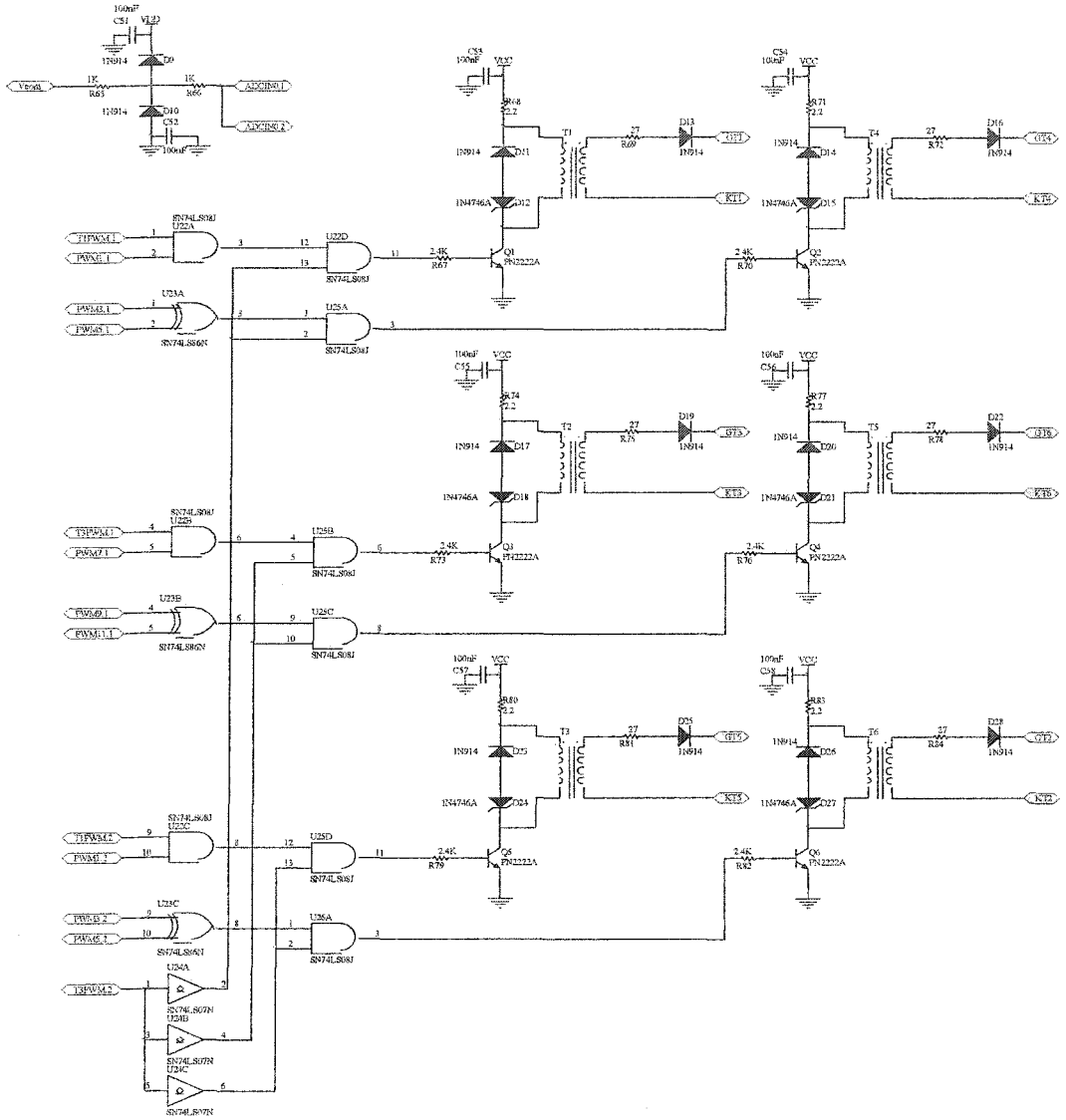


Figure F.3 : Circuit d'attaque des thyristors